DRAFT

Common Criteria

Common Criteria for Information Technology Security Evaluation

Part 3 : Security assurance requirements

3 April 1998

Version 2.0 Semi-Final

CCIB-98-18

DRAFT

Foreword

The CC Project Sponsoring Organisations are pleased to provide this **version 2.0 semi-final** of the *Common Criteria for Information Technology Security Evaluation*. This version is being submited by the CC project to ISO/IEC, JTC 1, SC27/WG3 for consideration as a draft international standard. The final release of CC version 2.0 will be prepared jointly with ISO representatives at the April 1998 meeting of SC27/WG3 in Stockholm, Sweden.

CANADA:

Communications Security Establishment
Criteria Coordinator
I2A Computer and Network Security
P.O. Box 9703, Terminal
Ottawa, Canada K1G 3Z4
Tel: +1.613.991.7882, Fax: +1.613.991.7455
E-mail: criteria@cse-cst.gc.ca
WWW: http://www.cse-cst.gc.ca/cse/english/cc.html
FTP: ftp://ftp.cse-cst.gc.ca/pub/criteria/CC2.0

FRANCE:

Service Central de la Sécurité des Systèmes d'Information (SCSSI)
Centre de Certification de la Sécurité des Technologies de l'Information
18, rue du docteur Zamenhof
F-92131 Issy les Moulineaux
France
Tel: +33.1.41463784, Fax: +33.1.41463701
E-mail: ssi20@calva.net

GERMANY:

Bundesamt für Sicherheit in der Informationstechnik (BSI)
German Information Security Agency (GISA)
Abteilung V

Postfach 20 03 63
D-53133 Bonn
Germany
Tel: +49.228.9582.300, Fax: +49.228.9582.427
E-mail: cc@bsi.de
WWW: http://www.bsi.bund.de

NETHERLANDS:

Netherlands National Communications Security Agency
P.O. Box 20061
NL 2500 EB The Hague
The Netherlands
Tel: +31.70.3485637, Fax: +31.70.3486503
E-mail: criteria@nlncsa.minbuza.nl
WWW: http://www.tno.nl/instit/fel/refs/cc.html

UNITED KINGDOM:

Communications-Electronics Security Group
Compusec Evaluation Methodology
P.O. Box 144
Cheltenham GL52 5UE
United Kingdom
Tel: +44.1242.221.491 ext. 5257, Fax: +44.1242.235.233
E-mail: criteria@cesg.gov.uk
WWW: http://www.cesg.gov.uk/cchtml
FTP: ftp://ftp.itsec.gov.uk/pub/ccv1.0

UNITED STATES - NIST:

National Institute of Standards and Technology
Computer Security Division
820 Diamond, MS: NN426
Gaithersburg, Maryland 20899
U.S.A.
Tel: +1.301.975.2934, Fax: +1.301.948.0279
E-mail: criteria@nist.gov
WWW: http://csrc.nist.gov/cc

UNITED STATES - NSA:

National Security Agency
Attn: V2, Common Criteria Technical Advisor

DRAFT

Fort George G. Meade, Maryland 20755-6740
U.S.A.
Tel: +1.410.859.4458, Fax: +1.410.684.7512
E-mail: common_criteria@radium.ncsc.mil
WWW: http://www.radium.ncsc.mil/tpep/

Table of Contents

D R A F T

DRAFT

D R A F T

List of Figures

DRAFT

List of Tables

1 Introduction

1.1 Scope

Part 3 defines the assurance requirements of the CC. It includes the evaluation assurance levels (EALs) that define a scale for measuring assurance, the individual assurance components from which the assurance levels are composed, and the criteria for evaluation of PPs and STs.

1.2 Organisation of Part 3

Chapter 1 is the introduction and paradigm for Part 3.

Chapter 2 describes the presentation structure of the assurance classes, families, components, and evaluation assurance levels along with their relationships. It also characterises the assurance classes and families found in Chapter 8 through 14.

Chapters 3, 4 and 5 provide a brief introduction to the evaluation criteria for PPs and STs, followed by detailed explanations of the families and components that are used for those evaluations.

Chapter 6 provides detailed definitions of the EALs.

Chapter 7 provides a brief introduction to the assurance classes and is followed by Chapters 8 through 14 that provide detailed definitions of those classes.

Chapters 15 and 16 provide a brief introduction to the evaluation criteria for maintenance of assurance, followed by detailed definitions of those families and components.

Annex A provides a summary of the dependencies between the assurance components.

Annex B provides a cross reference between the EALs and the assurance components.

Annex C provides the Common Criteria observation report guidance, example observations and example printed form.

1.3 CC assurance paradigm

The purpose of this section is to document the philosophy that underpins the CC approach to assurance. An understanding of this section will permit the reader to understand the rationale behind the CC assurance requirements.

1.3.1 CC philosophy

The CC philosophy is that the threats to security and organisational security policy commitments should be clearly articulated and the proposed security measures be demonstrably sufficient for their intended purpose.

Furthermore, measures should be adopted that reduce the likelihood of vulnerabilities, the ability to intentionally exploit or unintentionally trigger (i.e. exercise) a vulnerability, and the extent of the damage that could occur from a vulnerability being exercised. Additionally, measures should be adopted that facilitate the subsequent identification of vulnerabilities and the elimination, mitigation, and/or notification that a vulnerability has been exploited or triggered.

1.3.2 Assurance approach

The CC philosophy is to provide assurance based upon an evaluation (active investigation) of the IT product or system that is to be trusted. Evaluation has been the traditional means of providing assurance and is the basis for prior evaluation criteria documents. In aligning the existing approaches, the CC adopts the same philosophy. The CC proposes measuring the validity of the documentation and of the resulting IT product or system by expert evaluators with increasing emphasis on scope, depth, and rigour.

The CC does not exclude, nor does it comment upon, the relative merits of other means of gaining assurance. Research continues with respect to alternative ways of gaining assurance. As mature alternative approaches emerge from these research activities, they will be considered for inclusion in the Common Criteria, which is so structured as to allow their future introduction.

1.3.2.1 Significance of vulnerabilities

It is assumed that there are threat agents that will actively seek to exploit opportunities to violate security policies both for illicit gains and for well-intentioned, but nonetheless insecure actions. Threat agents may also accidentally trigger security vulnerabilities, causing harm to the organisation. Due to the need to process sensitive information and the lack of availability of sufficiently trusted products or systems, there is significant risk due to failures of IT. It is, therefore, likely that IT security breaches could lead to significant loss.

IT security breaches arise through the intentional exploitation or the unintentional triggering of vulnerabilities in the application of IT within business concerns.

Steps should be taken to prevent vulnerabilities arising in IT products and systems. To the extent feasible, vulnerabilities should be:

a) eliminated — that is, active steps should be taken to expose, and remove or neutralise, all exercisable vulnerabilities;

b) minimised — that is, active steps should be taken to reduce, to an acceptable residual level, the potential impact of any exercise of a vulnerability;

c) monitored — that is, active steps should be taken to ensure that any attempt to exercise a residual vulnerability will be detected so that steps can be taken to limit the damage.

1.3.2.2 Cause of vulnerabilities

Vulnerabilities can arise through failures in:

a) requirements — that is, an IT product or system may possess all the functions and features required of it and still contain vulnerabilities that render it unsuitable or ineffective with respect to security;

b) construction — that is, an IT product or system does not meet its specifications and/or vulnerabilities have been introduced as a result of poor constructional standards or incorrect design choices;

c) operation — that is, an IT product or system has been constructed correctly to a correct specification but vulnerabilities have been introduced as a result of inadequate controls upon the operation.

1.3.2.3 CC assurance

Assurance is grounds for confidence that an IT product or system meets its security objectives. Assurance can be derived from reference to sources such as unsubstantiated assertions, prior relevant experience, or specific experience. However, the CC provides assurance through active investigation. Active investigation is an evaluation of the IT product or system in order to determine its security properties.

1.3.2.4 Assurance through evaluation

Evaluation has been the traditional means of gaining assurance, and is the basis of the CC approach. Evaluation techniques can include, but are not limited to:

a) analysis and checking of process(es) and procedure(s);

b) checking that process(es) and procedure(s) are being applied;

c) analysis of the correspondence between TOE design representations;

d) analysis of the TOE design representation against the requirements;

e) verification of mathematical proofs;

f) analysis of guidance documents;

g) analysis of functional tests developed and the results provided;

h) independent functional testing;

i) analysis for vulnerabilities (including flaw hypothesis);

j) penetration testing.

1.3.3 The CC evaluation assurance scale

The CC philosophy assumes that greater assurance results from the application of greater evaluation effort, and that the goal is to apply the minimum effort required to provide the necessary level of assurance. The increasing level of effort is based upon:

a) scope — that is, the effort is greater because a larger portion of the IT product or system is included;

b) depth — that is, the effort is greater because it is deployed to a finer level of design and implementation detail;

c) rigour — that is, the effort is greater because it is applied in a more structured, formal manner.

2 Security assurance requirements

2.1 Structures

The following sections describe the constructs used in representing the assurance classes, families, components, and EALs along with the relationships among them.

Figure 2.1 illustrates the assurance requirements defined in Part 3 of the CC. Note that the most abstract collection of assurance requirements is referred to as a class. Each class contains assurance families, which then contain assurance components, which in turn contain assurance elements. Classes and families are used to provide a taxonomy for classifying assurance requirements, while components are used to specify assurance requirements in a PP/ST.

2.1.1 Class structure

Figure 2.1 illustrates the assurance class structure.

2.1.1.1 Class name

Each assurance class is assigned a unique name. The name indicates the topics covered by the assurance class.

A unique short form of the assurance class name is also provided. This is the primary means for referencing the assurance class. The convention adopted is an “A” followed by two letters related to the class name.

2.1.1.2 Class introduction

Each assurance class has an introductory section that describes the composition of the class and contains supportive text covering the intent of the class.

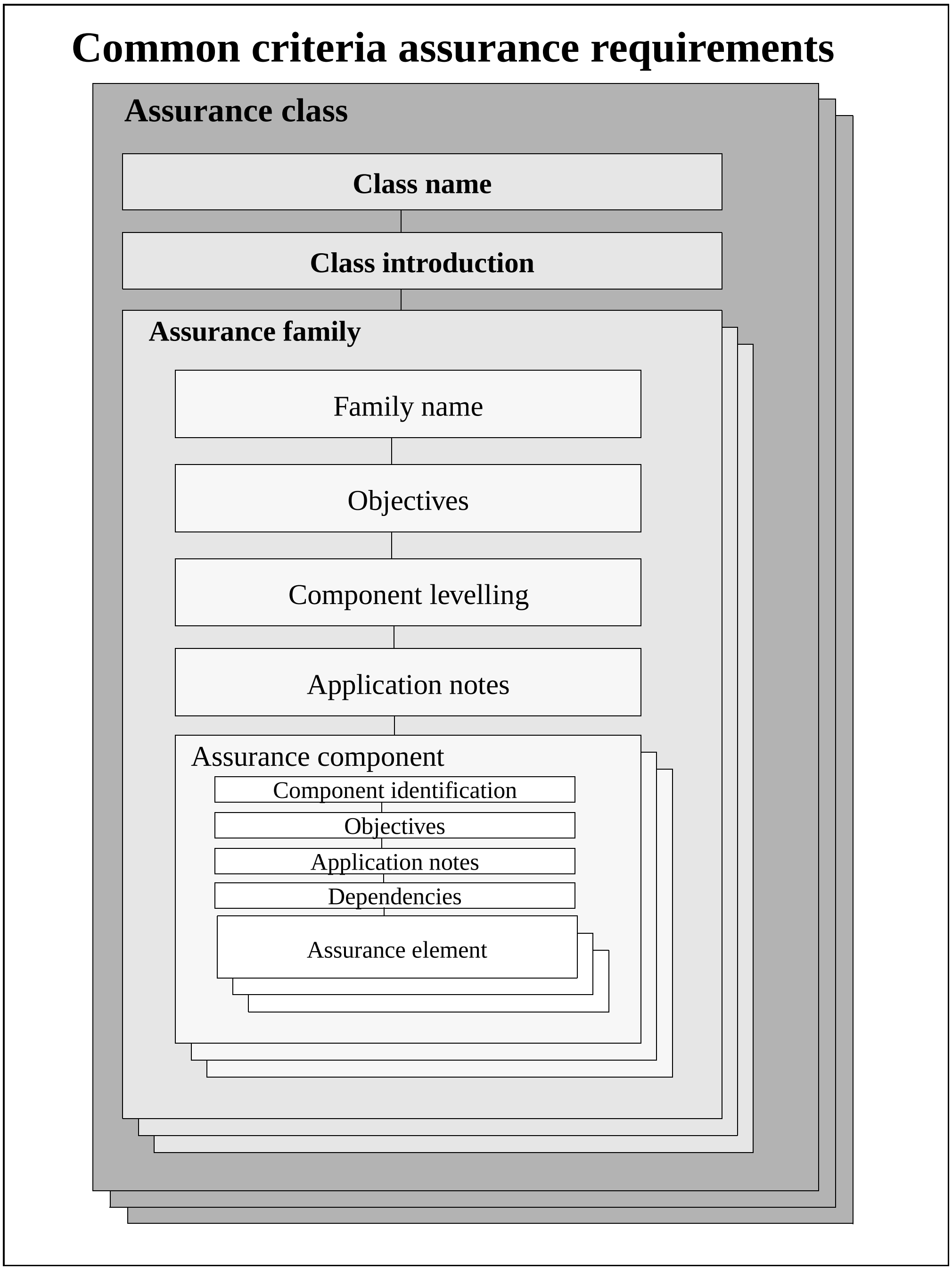

Figure 2.1 - Assurance class/family/component/element hierarchy

2.1.1.1 Assurance families

Each assurance class contains at least one assurance family. The structure of the assurance families is described in the following section.

2.1.2 Assurance family structure

Figure 2.1 illustrates the assurance family structure.

2.1.2.1 Family name

Every assurance family is assigned a unique name. The name provides descriptive information about the topics covered by the assurance family. Each assurance family is placed within the assurance class that contains other families with the same intent.

A unique short form of the assurance family name is also provided. This is the primary means used to reference the assurance family. The convention adopted is that the short form of the class name is used, followed by an underscore, and then three letters related to the family name.

2.1.2.2 Objectives

The objectives section of the assurance family presents the intent of the assurance family.

This section describes the objectives, particularly those related to the CC assurance paradigm, that the family is intended to address. The description for the assurance family is kept at a general level. Any specific details required for objectives are incorporated in the particular assurance component.

2.1.2.3 Component levelling

Each assurance family contains one or more assurance components. This section of the assurance family describes the components available and explains the distinctions between them. Its main purpose is to differentiate between the assurance components once it has been determined that the assurance family is a necessary or useful part of the assurance requirements for a PP/ST.

Assurance families containing more than one component are levelled and rationale is provided as to how the components are levelled. This rationale is in terms of scope, depth, and/or rigour.

2.1.2.4 Application notes

The application notes section of the assurance family, if present, contains additional information for the assurance family. This information should be of particular interest to users of the assurance family (e.g. PP and ST authors, designers of TOEs, evaluators). The presentation is informal and covers, for example, warnings about limitations of use and areas where specific attention may be required.

2.1.2.5 Assurance components

Each assurance family has at least one assurance component. The structure of the assurance components is provided in the following section.

2.1.3 Assurance component structure

Figure 2.2 illustrates the assurance component structure.

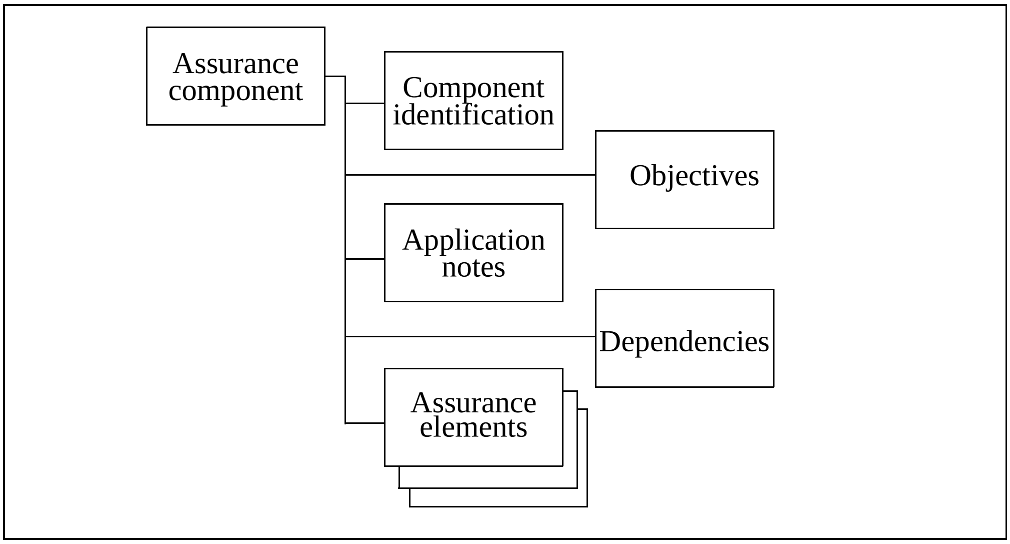

Figure 2.2 - Assurance component structure

The relationship between components within a family is highlighted using a bolding convention. Those parts of the requirements that are new, enhanced or modified beyond the requirements of the previous component within a hierarchy are bolded. The same bolding convention is also used for dependencies.

2.1.3.1 Component identification

The component identification section provides descriptive information necessary to identify, categorise, register, and reference a component.

Every assurance component is assigned a unique name. The name provides descriptive information about the topics covered by the assurance component. Each assurance component is placed within the assurance family that shares its security objective.

A unique short form of the assurance component name is also provided. This is the primary means used to reference the assurance component. The convention used is that the short form of the family name is used, followed by a period, and then a numeric character. The numeric characters for the components within each family are assigned sequentially, starting from 1.

2.1.3.2 Objectives

The objectives section of the assurance component, if present, contains specific objectives for the particular assurance component. For those assurance components that have this section, it presents the specific intent of the component and a more detailed explanation of the objectives.

2.1.3.3 Application notes

The application notes section of an assurance component, if present, contains additional information to facilitate the use of the component.

2.1.3.4 Dependencies

Dependencies among assurance components arise when a component is not self-sufficient, and relies upon the presence of another component.

Each assurance component provides a complete list of dependencies to other assurance components. Some components may list “No dependencies”, to indicate that no dependencies have been identified. The components depended upon may have dependencies on other components.

The dependency list identifies the minimum set of assurance components which are relied upon. Components which are hierarchical to a component in the dependency list may also be used to satisfy the dependency.

In specific situations the indicated dependencies might not be applicable. The PP/ST author, by providing rationale for why a given dependency is not applicable, may elect not to satisfy that dependency.

2.1.3.5 Assurance elements

A set of assurance elements is provided for each assurance component. An assurance element is a security requirement which, if further divided, would not yield a meaningful evaluation result. It is the smallest security requirement recognised in the CC.

Each assurance element is identified as belonging to one of the three sets of assurance elements:

a) Developer action elements: the activities that shall be performed by the developer. This set of actions is further qualified by evidential material referenced in the following set of elements. Requirements for developer actions are identified by appending the letter “D” to the element number.

b) Content and presentation of evidence elements: the evidence required, what the evidence shall demonstrate, and what information the evidence shall convey. Requirements for content and presentation of evidence are identified by appending the letter “C” to the element number.

c) Evaluator action elements: the activities that shall be performed by the evaluator. This set of actions implicitly includes confirmation that the requirements prescribed in the previous two sets of elements have been met, and includes actions or analysis that shall be performed in addition to that already performed by the developer. Requirements for evaluator actions are identified by appending the letter “E” to the element number.

The developer actions and content and presentation of evidence define the assurance requirements that are used to represent a developer’s responsibilities in demonstrating assurance in the TOE security functions. By meeting these requirements, the developer can increase confidence that the TOE satisfies the functional and assurance requirements of a PP or a ST.

The evaluator actions define the evaluator's responsibilities in the two aspects of evaluation. The first aspect is validation of the PP/ST, in accordance with the classes APE and ASE in chapters 4 and 5. The second aspect is verification of the TOE's conformance with its functional and assurance requirements. By demonstrating that the PP/ST is valid and that the requirements are met by the TOE, the evaluator can provide a basis for confidence that the TOE will meet its security objectives.

Evaluator actions, combined with the requirements for content and presentation of evidence, identify the evaluator effort that shall be expended in verifying the security claims made in the ST of the TOE.

2.1.4 Assurance elements

Each element represents a requirement to be met. These statements of requirements are intended to be clear, concise, and unambiguous. Therefore, there are no compound sentences: each separable requirement is stated as an individual element.

The elements have been written using the normal dictionary meaning for the terms used, rather than using a number of predefined terms as shorthand which results in implicit requirements. Therefore, elements are written as explicit requirements, with *no reserved terms*.

In contrast to Part 2, neither assignment nor selection operations are relevant for elements in Part 3 of the CC; however, refinements may be made to Part 3 elements as required.

2.1.5 EAL structure

Figure 2.3 illustrates the EALs and associated structure defined in this Part 3. Note that while the figure shows the contents of the assurance components, it is intended that this information would be included in an EAL by reference to the actual components defined in the CC.

2.1.5.1 EAL name

Each EAL is assigned a unique name. The name provides descriptive information about the intent of the EAL.

A unique short form of the EAL name is also provided. This is the primary means used to reference the EAL.

2.1.5.2 Objectives

The objectives section of the EAL presents the intent of the EAL.

2.1.5.3 Application notes

The application notes section of the EAL, if present, contains information of particular interest to users of the EAL (e.g. PP and ST authors, designers of TOEs targeting this EAL, evaluators). The presentation is informal and covers, for example, warnings about limitations of use and areas where specific attention may be required.

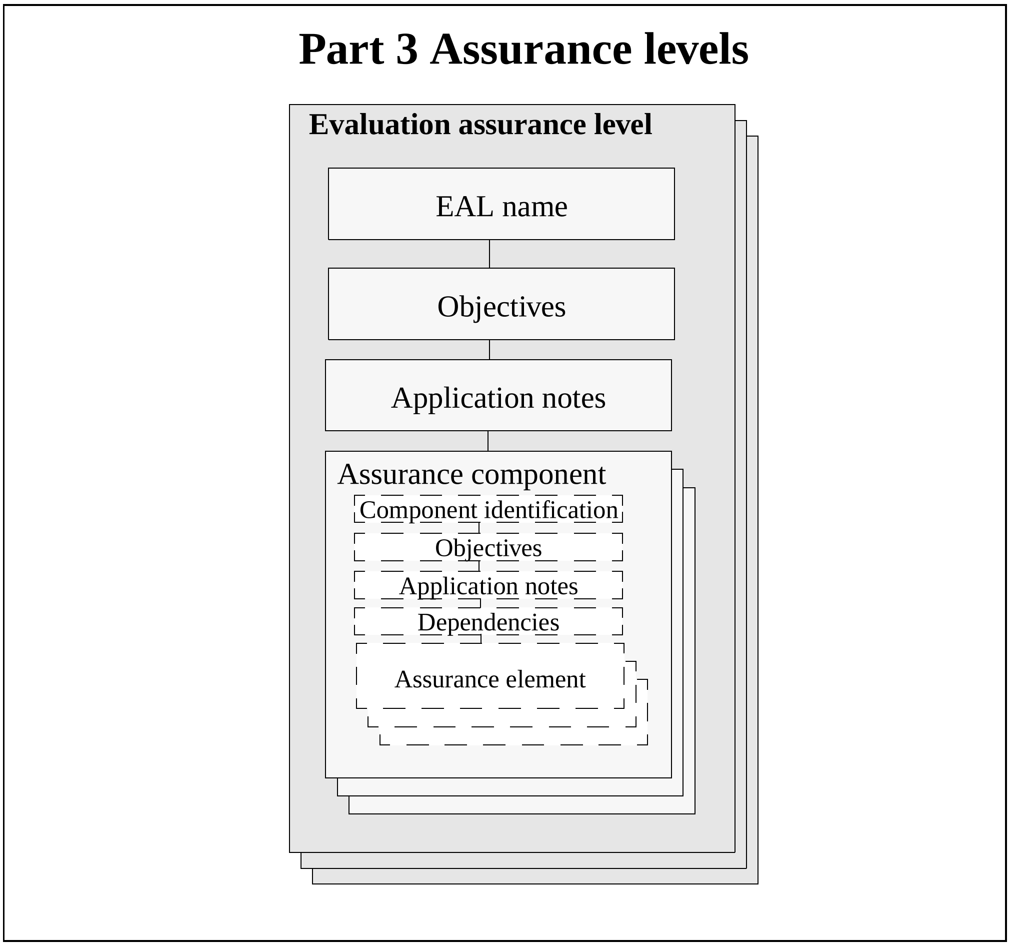

Figure 2.3 - EAL structure

D R A F T

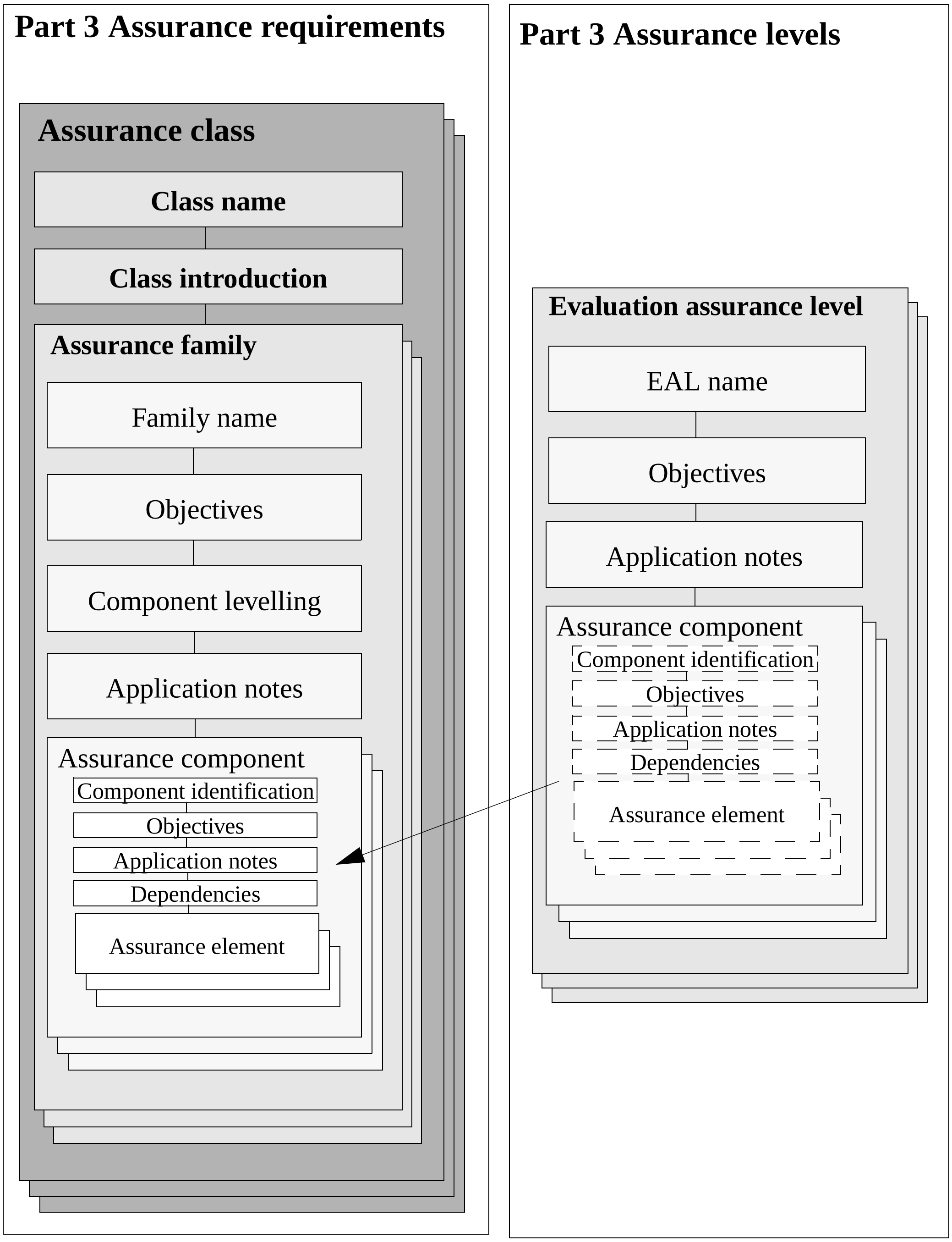

Figure 2.4 - Assurance and assurance level association

2.1.5.1 Assurance components

A set of assurance components have been chosen for each EAL.

A higher level of assurance than that provided by a given EAL can be achieved by:

a) including additional assurance components from other assurance families; or

b) replacing an assurance component with a higher level assurance component from the same assurance family.

2.1.6 Relationship between assurances and assurance levels

Figure 2.4 illustrates the relationship between the assurance requirements and the assurance levels defined in Part 3. While assurance components further decompose into assurance elements, assurance elements cannot be individually referenced by assurance levels. Note that the arrow in the figure represents a reference from an EAL to an assurance component within the class where it is defined.

2.2 Component taxonomy

This Part 3 contains classes of families and components that are grouped on the basis of related assurance. At the start of each class is a diagram that indicates the families in the class and the components in each family.

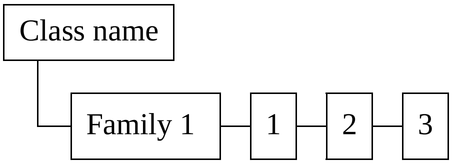

Figure 2.5 - Sample class decomposition diagram

In Figure 2.5, above, the class as shown contains a single family. The family contains three components that are linearly hierarchical (i.e. component 2 requires more than component 1, in terms of specific actions, specific evidence, or rigour of the actions or evidence). The assurance families in this Part 3 are all linearly hierarchical, although linearity is not a mandatory criterion for assurance families that may be added in the future.

2.3 Protection Profile and Security Target evaluation criteria class structure

The requirements for protection profile and security target evaluation are treated as assurance classes and are presented using the similar structure as that used for the other assurance classes, described below. One notable difference is the absence of a component levelling section in the associated family descriptions. The reason is that each family has only a single component and therefore no levelling has occurred.

Tables 3.1, 3.2, 3.3 and 3.2 in Chapter 3 of this Part 3 summarise, for both the APE and ASE classes, their constituent families and abbreviations for each. Narrative summaries for the APE families can be found in Part 1, Annex B, sections B.2.2 through B.2.6, whereas narrative summaries for the ASE families can be found in Part 1, Annex C, sections C.2.2 through C.2.8.

2.4 Usage of terms in Part 3

The following is a list of terms which are used in a precise way in Part 3. They do not merit inclusion in the glossary because they are general English terms and their usage, though restricted to the explanations given below, is in conformance with dictionary definitions. However, those explanations of the terms were used as guidance in the development of Part 3 and should be helpful for general understanding.

Check — This term is similar to, but less rigourous than “confirm” or “verify”. This term requires a quick determination to be made by the evaluator, perhaps requiring only a cursory analysis, or perhaps no analysis at all.

Coherent — An entity is logically ordered and with a discernible meaning. For documentation, this addresses both the actual text and the structure of the document, in terms of whether it is understandable by its target audience.

Complete — All necessary parts of an entity have been provided. In terms of documentation, this means that all relevant information is covered in the documentation, at such a level of detail that no further explanation is required at that level of abstraction.

Confirm — This term is used to indicate that something needs to be reviewed in detail, and that an independent determination of sufficiency needs to be made. The level of rigour required depends on the nature of the subject matter. This term is only applied to evaluator actions.

Consistent — This term describes a relationship between two or more entities, indicating that there are no apparent contradictions between these entities.

Counter (verb) — This term is typically used in the context that a security objective counters a particular threat, but does not necessarily indicate that the threat is completely eradicated as a result.

Demonstrate — This term refers to an analysis leading to a conclusion, which is less rigourous than a “proof”.

Describe — This term requires that certain, specific details of an entity be provided.

Determine — This term requires an independent analysis to be made, with the objective of reaching a particular conclusion. The usage of this term differs from “confirm” or “verify”, since these other terms imply that an analysis has already been performed which needs to be reviewed, whereas the usage of “determine” implies a truly independent analysis, usually in the absence of any previous analysis having been performed.

Ensure — This term, used by itself, implies a strong causal relationship between an action and its consequences. This term is typically preceded by the word “helps”, which indicates that the consequence is not fully certain, on the basis of that action alone.

Exhaustive — This term is used in the CC with respect to conducting an analysis or other activity. It is related to “systematic” but is considerably stronger, in that it indicates not only that a methodical approach has been taken to perform the analysis or activity according to an unambiguous plan, but that the plan that was followed is sufficient to ensure that all possible avenues have been exercised.

Explain — This term differs from both “describe” and “demonstrate”. It is intended to answer the question “Why?” without actually attempting to argue that the course of action that was taken was necessarily optimal.

Internally consistent — There are no contradictions between any aspects of an entity. In terms of documentation, this means that there can be no statements within the documentation that can be taken to contradict each other.

Justification — This term refers to an analysis leading to a conclusion, but is more rigorous than a demonstration. This term requires significant rigour in terms of very carefully and thoroughly explaining every step of a logical argument.

Mutually supportive — This term describes a relationship between a group of entities, indicating that the entities possess properties which do not conflict with, and may assist the other entities in performing their tasks. It is not necessary to determine that every individual entity in question directly supports other entities in that grouping; rather, it is a more general determination that is made.

Prove — This refers to a formal analysis in its mathematical sense. It is completely rigourous in all ways. Typically, “prove” is used when there is a desire to show correspondence between two TSF representations at a high level of rigour.

Specify — This term is used in the same context as “describe”, but is intended to be more rigourous and precise. It is very similar to “define”.

Trace (verb) — This term is used to indicate that an informal correspondence is required between two entities with only a minimal level of rigour.

Verify — This term is similar in context to “confirm”, but has more rigourous connotations. This term when used in the context of evaluator actions indicates that an independent effort is required of the evaluator.

2.5 Assurance categorisation

The assurance classes, families, and the abbreviation for each family are shown in Table 2.1.

2.6 Assurance class and family overview

The following summarises the assurance classes and families of Chapter 8-14. These classes and family summaries are presented in the same order as they appear in Chapters 8-14.

2.6.1 Class ACM: Configuration management

Configuration management (CM) helps to ensure that the integrity of the TOE is preserved, by requiring discipline and control in the processes of refinement and modification of the TOE. CM

Assurance Class	Assurance Family	Abbreviated Name
Class ACM: Configuration management	CM automation	ACM_AUT
	CM capabilities	ACM_CAP
	CM scope	ACM_SCP
Class ADO: Delivery and operation	Delivery	ADO_DEL
	Installation, generation and start-up	ADO_IGS
Class ADV: Development	Functional specification	ADV_FSP
	High-level design	ADV_HLD
	Implementation representation	ADV_IMP
	TSF internals	ADV_INT
	Low-level design	ADV_LLD
	Representation correspondence	ADV_RCR
	Security policy modeling	ADV_SPM
Class AGD: Guidance documents	Administrator guidance	AGD_ADM
	User guidance	AGD_USR
Class ALC: Life cycle support	Development security	ALC_DVS
	Flaw remediation	ALC_FLR
	Life cycle definition	ALC_LCD
	Tools and techniques	ALC_TAT
Class ATE: Tests	Coverage	ATE_COV
	Depth	ATE_DPT
	Functional tests	ATE_FUN
	Independent testing	ATE_IND
Class AVA: Vulnerability assessment	Covert channel analysis	AVA_CCA
	Misuse	AVA_MSU
	Strength of TOE security functions	AVA_SOF
	Vulnerability analysis	AVA_VLA

Table 2.1 -Assurance family breakdown and mapping

prevents unauthorised modifications, additions, or deletions to the TOE, thus providing assurance that the TOE and documentation used for evaluation are the ones prepared for distribution.

2.6.1.1 CM automation (ACM_AUT)

Configuration management automation establishes the level of automation used to control the configuration items.

2.6.1.2 CM capabilities (ACM_CAP)

Configuration management capabilities define the characteristics of the configuration management system.

2.6.1.3 CM scope (ACM_SCP)

Configuration management scope indicates the TOE items that need to be controlled by the configuration management system.

2.6.2 Class ADO: Delivery and operation

Assurance class ADO defines requirements for the measures, procedures, and standards concerned with secure delivery, installation, and operational use of the TOE, ensuring that the security protection offered by the TOE is not compromised during transfer, installation, start-up, and operation.

2.6.2.1 Delivery (ADO_DEL)

Delivery covers the procedures used to maintain security during transfer of the TOE to the user, both on initial delivery and as part of subsequent modification. It includes special procedures or operations required to demonstrate the authenticity of the delivered TOE. Such procedures and measures are the basis for ensuring that the security protection offered by the TOE is not compromised during transfer. While compliance with the delivery requirements cannot always be determined when a TOE is evaluated, it is possible to evaluate the procedures that a developer has developed to distribute the TOE to users.

2.6.2.2 Installation, generation and start-up (ADO_IGS)

Installation, generation, and start-up requires that the copy of the TOE is configured and activated by the administrator to exhibit the same protection properties as the master copy of the TOE. The installation, generation, and start-up procedures provide confidence that the administrator will be aware of the TOE configuration parameters and how they can affect the TSF.

2.6.3 Class ADV: Development

Assurance class ADV defines requirements for the stepwise refinement of the TSF from the TOE summary specification in the ST down to the actual implementation. Each of the resulting TSF representations provide information to help the evaluator determine whether the functional requirements of the TOE have been met.

2.6.3.1 Functional specification (ADV_FSP)

The functional specification describes the TSF, and must be a complete and accurate instantiation of the TOE security functional requirements. The functional specification also details the external interface to the TOE. Users of the TOE are expected to interact with the TSF through this interface.

2.6.3.2 High-level design (ADV_HLD)

The high-level design is a top level design specification that refines the TSF functional specification into the major constituent parts of the TSF. The high level design identifies the basic structure of the TSF and the major hardware, firmware, and software elements.

2.6.3.3 Implementation representation (ADV_IMP)

The implementation representation is the least abstract representation of the TSF. It captures the detailed internal workings of the TSF in terms of source code, hardware drawings, etc., as applicable.

2.6.3.4 TSF internals (ADV_INT)

The TSF internals requirements specify the requisite internal structuring of the TSF.

2.6.3.5 Low-level design (ADV_LLD)

The low-level design is a detailed design specification that refines the high-level design into a level of detail that can be used as a basis for programming and/or hardware construction.

2.6.3.6 Representation correspondence (ADV_RCR)

The representation correspondence is a demonstration of mappings between all adjacent pairs of available TSF representations, from the TOE summary specification through to the least abstract TSF representation that is provided.

2.6.3.7 Security policy modeling (ADV_SPM)

Security policy models are structured representations of security policies of the TSP, and are used to provide increased assurance that the functional specification corresponds to the security policies of the TSP, and ultimately to the TOE security functional requirements. This is achieved via correspondence mappings between the functional specification, the security policy model, and the security policies that are modelled.

2.6.4 Class AGD: Guidance documents

Assurance class AGD defines requirements directed at the understandability, coverage and completeness of the operational documentation provided by the developer. This documentation, which provides two categories of information, for end users and for administrators, is an important factor in the secure operation of the TOE.

2.6.4.1 Administrator guidance (AGD_ADM)

Requirements for administrative guidance help ensure that the environmental constraints can be understood by administrators and operators of the TOE. Administrative guidance is the primary means available to the developer for providing the TOE administrators with detailed, accurate information of how to administer the TOE in a secure manner and how to make effective use of the TSF privileges and protection functions.

2.6.4.2 User guidance (AGD_USR)

Requirements for user guidance help ensure that users are able to operate the TOE in a secure manner (e.g. the usage constraints assumed by the PP or ST must be clearly explained and illustrated). User guidance is the primary vehicle available to the developer for providing the TOE users with the necessary background and specific information on how to correctly use the TOE's protection functions. User guidance must do two things. First, it needs to explain what the user-visible security functions do and how they are to be used, so that users are able to consistently and effectively protect their information. Second, it needs to explain the user's role in maintaining the TOE's security.

2.6.5 Class ALC: Life cycle support

Assurance class ALC defines requirements for assurance through the adoption of a well defined life-cycle model for all the steps of the TOE development, including flaw remediation procedures and policies, correct use of tools and techniques and the security measures used to protect the development environment.

2.6.5.1 Development security (ALC_DVS)

Development security covers the physical, procedural, personnel, and other security measures used in the development environment. It includes physical security of the development location(s) and controls on the selection and hiring of development staff.

2.6.5.2 Flaw remediation (ALC_FLR)

Flaw remediation ensures that flaws discovered by the TOE consumers will be tracked and corrected while the TOE is supported by the developer. While compliance with the flaw remediation requirements cannot be determined when a TOE is evaluated, it is possible to evaluate the procedures and policies that a developer has in place to track and repair flaws, and to distribute the repairs to consumers.

2.6.5.3 Life cycle definition (ALC_LCD)

Life cycle definition establishes that the engineering practices used by a developer to produce the TOE include the considerations and activities identified in the development process and operational support requirements. Confidence in the correspondence between the requirements and the TOE is greater when security analysis and the production of evidence are done on a regular basis as an integral part of the development process and operational support activities. It is not the intent of this component to dictate any specific development process.

2.6.5.4 Tools and techniques (ALC_TAT)

Tools and techniques addresses the need to define the development tools being used to analyse and implement the TOE. It includes requirements concerning the development tools and implementation dependent options of those tools.

2.6.6 Class ATE: Tests

Assurance class ATE states testing requirements that demonstrate that the TSF satisfies the TOE security functional requirements.

2.6.6.1 Coverage (ATE_COV)

Coverage deals with the completeness of the functional tests performed by the developer on the TOE. It addresses the extent to which the TOE security functions are tested.

2.6.6.2 Depth (ATE_DPT)

Depth deals with the level of detail to which the developer tests the TOE. Testing of security functions is based upon increasing depth of information derived from analysis of the representations.

2.6.6.3 Functional tests (ATE_FUN)

Functional testing establishes that the TSF exhibits the properties necessary to satisfy the requirements of its ST. Functional testing provides assurance that the TSF satisfies at least the requirements of the chosen functional components. However, functional tests do not establish that the TSF does no more than expected. This family focuses on functional testing performed by the developer.

2.6.6.4 Independent testing (ATE_IND)

Independent testing specifies the degree to which the functional testing of the TOE must be performed by a party other than the developer (e.g. a third party). This family adds value by the introduction of tests that are not part of the developers tests.

2.6.7 Class AVA: Vulnerability assessment

Assurance class AVA defines requirements directed at the identification of exploitable vulnerabilities. Specifically, it addresses those vulnerabilities introduced in the construction, operation, misuse, or incorrect configuration of the TOE.

2.6.7.1 Covert channel analysis (AVA_CCA)

Covert channel analysis is directed towards the discovery and analysis of unintended communications channels that can be exploited to violate the intended TSP.

2.6.7.2 Misuse (AVA_MSU)

Misuse analysis investigates whether an administrator or end-user, with an understanding of the guidance documentation, would reasonably be able to determine if the TOE is configured and operating in a manner that is insecure.

2.6.7.3 Strength of TOE security functions (AVA_SOF)

Strength of function analysis addresses TOE security functions that are realised by a probabilistic or permutational mechanism (e.g. a password or hash function). Even if such functions cannot be bypassed, deactivated, or corrupted, it may still be possible to defeat them by direct attack. A level or a specific metric may be claimed for the strength of each of these functions. Strength of function analysis is performed to determine whether such functions meet or exceed the claim. For example, strength of function analysis of a password mechanism can demonstrate whether the password function meets the strength claim by showing that the password space is sufficiently large.

2.6.7.4 Vulnerability analysis (AVA_VLA)

Vulnerability analysis consists of the identification of flaws potentially introduced in the different refinement steps of the development. It results in the definition of penetration tests through the collection of the necessary information concerning: (1) the completeness of the TSF (does the TSF counter all the postulated threats?) and (2) the dependencies between all security functions. These potential vulnerabilities are assessed through penetration testing to determine whether they could, in practice, be exploitable to compromise the security of the TOE.

2.7 Maintenance categorisation

The requirements for the maintenance of assurance are treated as an assurance class and are presented using the class structure defined above.

The maintenance of assurance families, and the abbreviation for each family are shown in Table 2.2.

Assurance Class	Assurance Family	Abbreviated Name
Maintenance of assurance	Assurance maintenance plan	AMA_AMP
	TOE component categorisation report	AMA_CAT
	Evidence of assurance maintenance	AMA_EVD
	Security impact analysis	AMA_SIA

Table 2.2 -Maintenance of assurance class decomposition

2.8 Maintenance of assurance class and family overview

The following summarises the assurance class and families of Chapter 16. The class and family summaries are presented in the same order as they appear in Chapter 16.

2.8.1 Class AMA: Maintenance of assurance

Class AMA: Maintenance of assurance is aimed at maintaining the level of assurance that the TOE will continue to meet its security target as changes are made to the TOE or its environment. Each of the families in this class identifies developer and evaluator actions that are to be applied *after* the TOE has been successfully evaluated, although some requirements can be applied at the time of the evaluation.

2.8.1.1 Assurance maintenance plan (AMA_AMP)

The assurance maintenance plan identifies the plans and procedures a developer must implement in order to ensure that the assurance that was established in the evaluated TOE is maintained as changes are made to the TOE or its environment.

2.8.1.2 TOE component categorisation report (AMA_CAT)

The TOE component categorisation report provides a categorisation of the components of a TOE (e.g. TSF subsystems) according to their relevance to security. This categorisation acts as a focus for the developer's security impact analysis.

2.8.1.3 Evidence of assurance maintenance (AMA_EVD)

Evidence of assurance maintenance seeks to establish confidence that the assurance is being maintained by the developer, in accordance with the assurance maintenance plan.

2.8.1.4 Security impact analysis (AMA_SIA)

Security impact analysis seeks to establish confidence that assurance has been maintained in the TOE through an analysis performed by the developer of the security impact of all changes affecting the TOE since it was evaluated.

3 Protection Profile and Security Target evaluation criteria

3.1 Overview

This chapter introduces the evaluation criteria for PPs and STs. The evaluation criteria are then fully presented in Chapter 4, Class APE: Protection Profile evaluation, and Chapter 5, Class ASE: Security Target evaluation.

These criteria are the first requirements presented in this part because the PP and ST evaluation will normally be performed before the TOE evaluation. They play a special role in that information about the TOE is assessed and the functional and assurance requirements are evaluated in order to find out whether the PP or ST is a meaningful basis for a TOE evaluation.

Although these evaluation criteria differ somewhat from the requirements in Chapters 7 through 14, they are presented in a similar manner because the developer and evaluator activities are comparable for PP, ST and TOE evaluations.

The PP and ST classes differ from the TOE classes in that all the requirements in the PP or ST class need to be considered for a PP or ST evaluation, whereas the requirements presented in the TOE classes cover a wide range of topics not all of which need be considered for a given TOE.

The evaluation criteria for PPs and STs are based on the information provided in Part 1, annexes B and C. Useful background information for the requirements in the classes APE and ASE, as presented in the following chapters, can be found there.

3.2 Protection Profile criteria overview

3.2.1 Protection Profile evaluation

The goal of a PP evaluation is to demonstrate that the PP is complete, consistent, technically sound, and hence suitable for use as a statement of requirements for one or more evaluatable TOEs. Such a PP may be eligible for inclusion within a PP registry.

3.2.2 Relation to the Security Target evaluation criteria

As described in Part 1, Annexes B and C, there are many similarities in structure and content between the generic PP and the TOE-specific ST. Consequently, the criteria for evaluating PPs contain requirements that are similar to many of those for STs, and the criteria for both are presented in a similar manner.

3.2.3 Evaluator tasks

3.2.3.1 Evaluator tasks for an evaluation based on CC requirements only

Evaluators performing a PP evaluation that does not include requirements from outside the CC shall apply the requirements of the APE class as described in Table 3.1.

Class	Family	Abbreviated Name
Class APE: Protection Profile evaluation	Protection Profile, TOE Description	APE_DES
	Protection Profile, Security Environment	APE_ENV
	Protection Profile, PP Introduction	APE_INT
	Protection Profile, Security Objectives	APE_OBJ
	Protection Profile, IT security requirements	APE_REQ

Table 3.1 -Protection Profile families - only CC requirements

3.2.3.2 Evaluator tasks for a CC extended evaluation

Evaluators performing a PP evaluation that includes requirements from outside the CC shall apply the requirements of the APE class as described in Table 3.2.

Class	Family	Abbreviated Name
Class APE: Protection Profile evaluation	Protection Profile, TOE Description	APE_DES
	Protection Profile, Security Environment	APE_ENV
	Protection Profile, PP Introduction	APE_INT
	Protection Profile, Security Objectives	APE_OBJ
	Protection Profile, IT security requirements	APE_REQ
	Protection Profile, Explicitly Stated IT Security Requirements	**APE_SRE**

Table 3.2 -Protection Profile families - CC extended requirements

3.3 Security Target criteria overview

3.3.1 Security Target evaluation

The goal of an ST evaluation is to demonstrate that the ST is complete, consistent, technically sound, and hence suitable for use as the basis for the corresponding TOE evaluation.

3.3.2 Relation to the other evaluation criteria in this Part

There are two identified stages for the evaluation of a TOE; the ST evaluation and the corresponding TOE evaluation. The requirements for ST evaluations are discussed here and in Chapter 6 while the requirements for TOE evaluations are contained in Chapters 7 through 14.

An ST evaluation includes a PP claims evaluation. If the ST does not claim PP conformance, the PP claims part of the ST shall contain a statement that the TOE does not claim conformance to any PP.

3.3.3 Evaluator tasks

3.3.3.1 Evaluator tasks for an evaluation based on CC requirements only

Evaluators performing an ST evaluation that does not include requirements from outside the CC shall apply the requirements of the ASE class as described in Table 3.3.

Class	Family	Abbreviated Name
Class ASE: Security Target evaluation	Security Target, TOE Description	ASE_DES
	Security Target, Security Environment	ASE_ENV
	Security Target, ST Introduction	ASE_INT
	Security Target, Security Objectives	ASE_OBJ
	Security Target, PP Claims	ASE_PPC
	Security Target, IT Security Requirements	ASE_REQ
	Security Target, TOE Summary Specification	ASE_TSS

Table 3.3 -Security Target families - only CC requirements

3.3.3.2 Evaluator tasks for a CC extended evaluation

Evaluators performing an ST evaluation that includes requirements from outside the CC shall apply the requirements of the ASE class as described in Table 3.4.

Class	Family	Abbreviated Name
Class ASE: Security Target evaluation	Security Target, TOE Description	ASE_DES
	Security Target, Security Environment	ASE_ENV
	Security Target, ST Introduction	ASE_INT
	Security Target, Security Objectives	ASE_OBJ
	Security Target, PP Claims	ASE_PPC
	Security Target, IT Security Requirements	ASE_REQ
	Security Target, Explicitly Stated IT Security Requirements	**ASE_SRE**
	Security Target, TOE Summary Specification	ASE_TSS

Table 3.4 -Security Target families - CC extended requirements

4 Class APE: Protection Profile evaluation

The goal of a PP evaluation is to demonstrate that the PP is complete, consistent and technically sound. An evaluated PP is suitable for use as the basis for the development of STs. Such a PP is eligible for inclusion in a registry.

Figure 4.1 shows the families within this class.

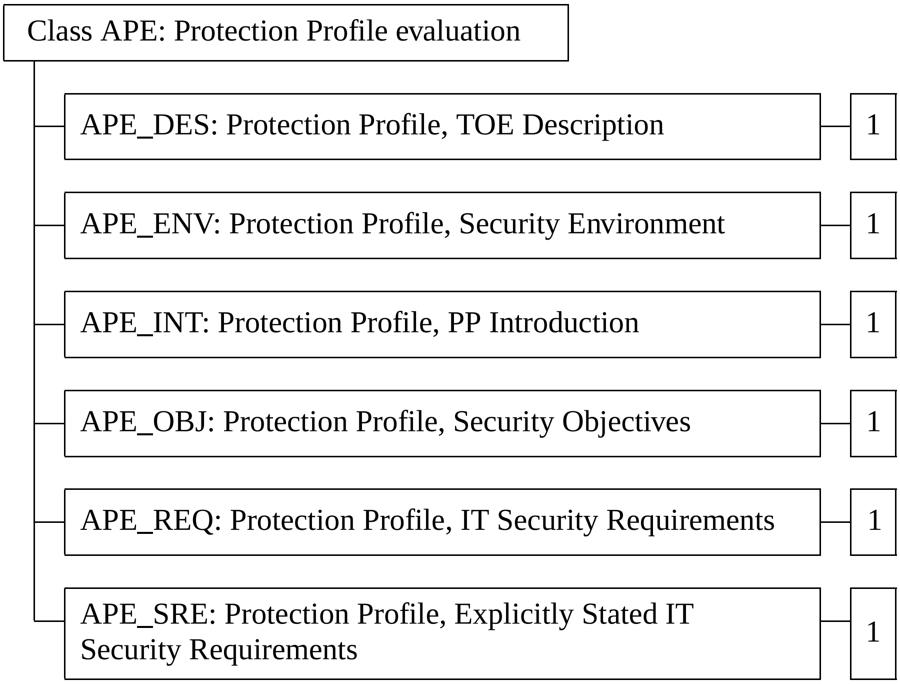
Class APE: Protection Profile evaluation
APE_DES: Protection Profile, TOE Description
1
APE_ENV: Protection Profile, Security Environment
1
APE_INT: Protection Profile, PP Introduction
1
APE_OBJ: Protection Profile, Security Objectives
1
APE_REQ: Protection Profile, IT Security Requirements
1
APE_SRE: Protection Profile, Explicitly Stated IT Security Requirements
1

Figure 4.1 - Protection Profile evaluation class decomposition

4.1 TOE description (APE_DES)

Objectives

The TOE description is an aid to the understanding of the TOE's security requirements. Evaluation of the TOE description is required to show that it is coherent, internally consistent and consistent with all other parts of the PP.

APE_DES.1 Protection Profile, TOE Description, Evaluation Requirements

Dependencies:

APE_ENV.1 Protection Profile, Security Environment, Evaluation Requirements

APE_INT.1 Protection Profile, PP Introduction, Evaluation Requirements

APE_OBJ.1 Protection Profile, Security Objectives, Evaluation Requirements

APE_REQ.1 Protection Profile, IT Security Requirements, Evaluation Requirements

Developer action elements:

APE_DES.1.1D **The PP developer shall provide a TOE Description as part of the PP.**

Content and presentation of evidence elements:

APE_DES.1.1C **The TOE description shall as a minimum describe the product type and the general IT features of the TOE.**

Evaluator action elements:

APE_DES.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

APE_DES.1.2E **The evaluator shall confirm that the TOE description is coherent and internally consistent.**

APE_DES.1.3E **The evaluator shall confirm that the TOE description is consistent with the other parts of the PP.**

4.2 Security environment (APE_ENV)

Objectives

In order to determine whether the IT security requirements in the PP are sufficient, it is important that the security problem to be solved is clearly understood by all parties to the evaluation.

APE_ENV.1 Protection Profile, Security Environment, Evaluation Requirements

Dependencies:

No dependencies.

Developer action elements:

APE_ENV.1.1D **The PP developer shall provide a statement of TOE security environment as part of the PP.**

Content and presentation of evidence elements:

APE_ENV.1.1C **The statement of TOE security environment shall identify and explain any assumptions about the intended usage of the TOE and the environment of use of the TOE.**

APE_ENV.1.2C **The statement of TOE security environment shall identify and explain any known or presumed threats to the assets against which protection will be required, either by the TOE or by its environment.**

APE_ENV.1.3C **The statement of TOE security environment shall identify and explain any organisational security policies with which the TOE must comply.**

Evaluator action elements:

APE_ENV.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

APE_ENV.1.2E **The evaluator shall confirm that the statement of TOE security environment is coherent and internally consistent.**

4.3 PP introduction (APE_INT)

Objectives

The PP introduction contains document management and overview information necessary to operate a PP registry. Evaluation of the PP introduction is required to demonstrate that the PP is correctly identified and that it is consistent with all other parts of the PP.

APE_INT.1 Protection Profile, PP Introduction, Evaluation Requirements

Dependencies:

APE_DES.1 Protection Profile, TOE Description, Evaluation Requirements

APE_ENV.1 Protection Profile, Security Environment, Evaluation Requirements

APE_OBJ.1 Protection Profile, Security Objectives, Evaluation Requirements

APE_REQ.1 Protection Profile, IT Security Requirements, Evaluation Requirements

Developer action elements:

APE_INT.1.1D **The PP developer shall provide a PP introduction as part of the PP.**

Content and presentation of evidence elements:

APE_INT.1.1C **The PP introduction shall contain a PP identification that provides the labelling and descriptive information necessary to identify, catalogue, register, and cross reference the PP.**

APE_INT.1.2C **The PP introduction shall contain a PP overview which summarises the PP in narrative form.**

Evaluator action elements:

APE_INT.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

APE_INT.1.2E **The evaluator shall confirm that the PP introduction is coherent and internally consistent.**

APE_INT.1.3E **The evaluator shall confirm that the PP introduction is consistent with the other parts of the PP.**

4.4 Security objectives (APE_OBJ)

Objectives

The security objectives is a concise statement of the intended response to the security problem. Evaluation of the security objectives is required to demonstrate that the stated objectives adequately address the security problem. The security objectives are categorised as security objectives for the TOE and as security objectives for the environment. The security objectives for both the TOE and the environment must be shown to be traced back to the identified threats to be countered and/or policies and assumptions to be met by each.

APE_OBJ.1 Protection Profile, Security Objectives, Evaluation Requirements

Dependencies:

APE_ENV.1 Protection Profile, Security Environment, Evaluation Requirements

Developer action elements:

APE_OBJ.1.1D **The PP developer shall provide a statement of security objectives as part of the PP.**

APE_OBJ.1.2D **The PP developer shall provide the security objectives rationale.**

Content and presentation of evidence elements:

APE_OBJ.1.1C **The statement of security objectives shall define the security objectives for the TOE and its environment.**

APE_OBJ.1.2C **The security objectives for the TOE shall be clearly stated and traced back to the identified threats to be countered by the TOE and/or organisational security policies to be met by the TOE.**

APE_OBJ.1.3C **The security objectives for the environment shall be clearly stated and traced back to aspects of identified threats not completely countered by the TOE and/ or organisational security policies or assumptions not completely met by the TOE.**

APE_OBJ.1.4C **The security objectives rationale shall demonstrate that the stated security objectives are suitable to counter the identified threats to security.**

APE_OBJ.1.5C **The security objectives rationale shall demonstrate that the stated security objectives are suitable to cover all of the identified organisational security policies and assumptions.**

Evaluator action elements:

APE_OBJ.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

APE_OBJ.1.2E **The evaluator shall confirm that the statement of security objectives is complete, coherent, and internally consistent.**

4.5 IT security requirements (APE_REQ)

Objectives

The IT security requirements chosen for a TOE and presented or cited in a PP need to be evaluated in order to confirm that they are internally consistent and lead to the development of a TOE that will meet its security objectives.

Not all of the IT security objectives expressed in a PP may be met by a compliant TOE, as some TOEs may depend on certain IT security requirements to be met by the IT environment. When this is the case, the environmental IT security requirements must be clearly stated and evaluated in context with the TOE requirements.

This family presents evaluation requirements that permit the evaluator to determine that a PP is suitable for use as a statement of requirements for an evaluatable TOE. The additional criteria necessary for the evaluation of explicitly stated requirements is covered in the APE_SRE family.

Application notes

The term “IT security requirements” refers to “TOE security requirements” and the optionally included “security requirements for the IT environment”.

The term “TOE security requirements” refers to “TOE security functional requirements” and/or “TOE security assurance requirements.”

In the APE_REQ.1 component, the word “appropriate” is used to indicate that certain elements allow options in certain cases. Which options are applicable depends on the given context in the PP. Detailed information for all these aspects is contained in Part 1, annex B.

APE_REQ.1 Protection Profile, IT Security Requirements, Evaluation Requirements

Dependencies:

APE_OBJ.1 Protection Profile, Security Objectives, Evaluation Requirements

Developer action elements:

APE_REQ.1.1D **The PP developer shall provide a statement of IT security requirements as part of the PP.**

APE_REQ.1.2D **The PP developer shall provide the security requirements rationale.**

Content and presentation of evidence elements:

APE_REQ.1.1C **The statement of TOE security functional requirements shall identify the TOE security functional requirements drawn from CC Part 2 functional requirements components.**

APE_REQ.1.2C **The statement of TOE security assurance requirements shall identify the TOE security assurance requirements drawn from CC Part 3 assurance requirements components.**

APE_REQ.1.3C **The statement of TOE security assurance requirements should include a CC Evaluation Assurance Level (EAL) as defined in CC Part 3.**

APE_REQ.1.4C **The evidence shall justify that the statement of TOE security assurance requirements is appropriate.**

APE_REQ.1.5C **The PP shall, if appropriate, identify any security requirements for the IT environment.**

APE_REQ.1.6C **All completed operations on IT security requirements included in the PP shall be identified.**

APE_REQ.1.7C **Any uncompleted operations on IT security requirements included in the PP shall be identified.**

APE_REQ.1.8C **Dependencies among the IT security requirements included in the PP should be satisfied.**

APE_REQ.1.9C **The evidence shall justify why any non-satisfaction of dependencies is appropriate.**

APE_REQ.1.10C **The PP shall include a statement of the minimum strength of function level for the TOE security functional requirements, either SOF-basic, SOF-medium or SOF-high, as appropriate.**

APE_REQ.1.11C **The PP shall identify any specific TOE security functional requirements for which an explicit strength of function is appropriate, together with the specific metric.**

APE_REQ.1.12C **The security requirements rationale shall demonstrate that the minimum strength of function level for the PP, together with any explicit strength of function claim, is consistent with the security objectives for the TOE.**

APE_REQ.1.13C **The security requirements rationale shall demonstrate that the IT security requirements are suitable to meet the security objectives.**

APE_REQ.1.14C **The security requirements rationale shall demonstrate that the set of IT security requirements together forms a mutually supportive and internally consistent whole.**

Evaluator action elements:

APE_REQ.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

APE_REQ.1.2E **The evaluator shall confirm that the statement of IT security requirements is complete, coherent, and internally consistent.**

4.6 Explicitly stated IT security requirements (APE_SRE)

Objectives

If, after careful consideration, none of the Part 2 or Part 3 requirements components are readily applicable to all or parts of the IT security requirements, the PP author may state other requirements which do not reference the CC. The use of such requirements shall be justified.

This family presents evaluation requirements that permit the evaluator to determine that the explicitly stated requirements are clearly and unambiguously expressed. The evaluation of requirements taken from the CC in conjunction with valid explicitly stated security requirements is addressed by the APE_REQ family.

Explicitly stated IT security requirements for a TOE presented or cited in a PP need to be evaluated in order to demonstrate that they are clearly and unambiguously expressed.

Application notes

Formulation of the explicitly stated requirements in a structure comparable to those of existing CC components and elements involves choosing similar labelling, manner of expression, and level of detail.

Using the CC requirements as a model means that the requirements can be clearly identified, that they are self-contained, and that the application of each requirement is feasible and will yield a meaningful evaluation result based on a compliance statement of the TOE for that particular requirement.

The term “IT security requirements” refers to “TOE security requirements” and the optionally included “security requirements for the IT environment”.

The term “TOE security requirements” refers to “TOE security functional requirements” and/or “TOE security assurance requirements.”

APE_SRE.1 Protection Profile, Explicitly Stated IT Security Requirements, Evaluation Requirements

Dependencies:

APE_REQ.1 Protection Profile, IT Security Requirements, Evaluation Requirements

Developer action elements:

APE_SRE.1.1D **The PP developer shall provide a statement of IT security requirements as part of the PP.**

APE_SRE.1.2D **The PP developer shall provide the security requirements rationale.**

Content and presentation of evidence elements:

APE_SRE.1.1C **All TOE security requirements that are explicitly stated without reference to the CC shall be identified.**

APE_SRE.1.2C **All security requirements for the IT environment that are explicitly stated without reference to the CC shall be identified.**

APE_SRE.1.3C **The evidence shall justify why the security requirements had to be explicitly stated.**

APE_SRE.1.4C **The explicitly stated IT security requirements shall use the CC requirements components, families and classes as a model for presentation.**

APE_SRE.1.5C **The explicitly stated IT security requirements shall be measurable and state objective evaluation requirements such that compliance or noncompliance of a TOE can be determined and systematically demonstrated.**

APE_SRE.1.6C **The explicitly stated IT security requirements shall be clearly and unambiguously expressed.**

APE_SRE.1.7C **The security requirements rationale shall demonstrate that the assurance requirements are applicable and appropriate to support any explicitly stated TOE security functional requirements.**

Evaluator action elements:

APE_SRE.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

APE_SRE.1.2E **The evaluator shall determine that all of the dependencies of the explicitly stated IT security requirements have been identified.**

5 Class ASE: Security Target evaluation

The goal of an ST evaluation is to demonstrate that the ST is complete, consistent, technically sound, and hence suitable for use as the basis for the corresponding TOE evaluation.

Figure 5.1 shows the families within this class.

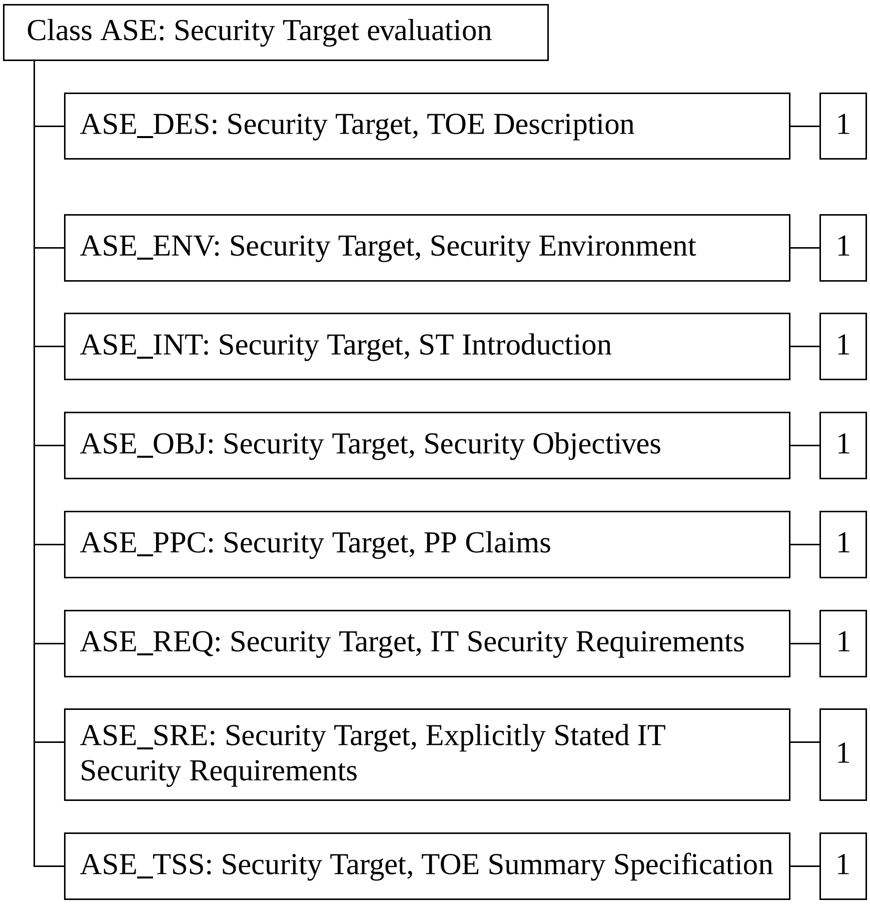
Class ASE: Security Target evaluation
ASE_DES: Security Target, TOE Description
1
ASE_ENV: Security Target, Security Environment
1
ASE_INT: Security Target, ST Introduction
1
ASE_OBJ: Security Target, Security Objectives
1
ASE_PPC: Security Target, PP Claims
1
ASE_REQ: Security Target, IT Security Requirements
1
ASE_SRE: Security Target, Explicitly Stated IT Security Requirements
1
ASE_TSS: Security Target, TOE Summary Specification
1

Figure 5.1 - Security Target evaluation class decomposition

5.1 TOE description (ASE_DES)

Objectives

The TOE description is an aid to the understanding of the TOE's security requirements. Evaluation of the TOE description is required to show that it is coherent, internally consistent and consistent with all other parts of the ST.

ASE_DES.1 Security Target, TOE Description, Evaluation Requirements

Dependencies:

ASE_ENV.1 Security Target, Security Environment, Evaluation Requirements

ASE_INT.1 Security Target, ST Introduction, Evaluation Requirements

ASE_OBJ.1 Security Target, Security Objectives, Evaluation Requirements

ASE_PPC.1 Security Target, PP Claims, Evaluation Requirements

ASE_REQ.1 Security Target, IT Security Requirements, Evaluation Requirements

ASE_TSS.1 Security Target, TOE Summary Specification, Evaluation Requirements

Developer action elements:

ASE_DES.1.1D **The ST developer shall provide a TOE Description as part of the ST.**

Content and presentation of evidence elements:

ASE_DES.1.1C **The TOE description shall as a minimum describe the product or system type, and the scope and boundaries of the TOE in general terms both in a physical and a logical way.**

Evaluator action elements:

ASE_DES.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ASE_DES.1.2E **The evaluator shall confirm that the TOE description is coherent and internally consistent.**

ASE_DES.1.3E **The evaluator shall confirm that the TOE description is consistent with the other parts of the ST.**

5.2 Security environment (ASE_ENV)

Objectives

In order to determine whether the IT security requirements in the ST are sufficient, it is important that the security problem to be solved is clearly understood by all parties to the evaluation.

ASE_ENV.1 Security Target, Security Environment, Evaluation Requirements

Dependencies:

No dependencies.

Developer action elements:

ASE_ENV.1.1D **The developer shall provide a statement of TOE security environment as part of the ST.**

Content and presentation of evidence elements:

ASE_ENV.1.1C **The statement of TOE security environment shall identify and explain any assumptions about the intended usage of the TOE and the environment of use of the TOE.**

ASE_ENV.1.2C **The statement of TOE security environment shall identify and explain any known or presumed threats to the assets against which protection will be required, either by the TOE or by its environment.**

ASE_ENV.1.3C **The statement of TOE security environment shall identify and explain any organisational security policies with which the TOE must comply.**

Evaluator action elements:

ASE_ENV.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ASE_ENV.1.2E **The evaluator shall confirm that the statement of TOE security environment is coherent and internally consistent.**

5.3 ST introduction (ASE_INT)

Objectives

The ST introduction contains identification and indexing material. Evaluation of the ST introduction is required to demonstrate that the ST is correctly identified and that it is consistent with all other parts of the ST.

ASE_INT.1 Security Target, ST Introduction, Evaluation Requirements

Dependencies:

ASE_DES.1 Security Target, TOE Description, Evaluation Requirements

ASE_ENV.1 Security Target, Security Environment, Evaluation Requirements

ASE_OBJ.1 Security Target, Security Objectives, Evaluation Requirements

ASE_PPC.1 Security Target, PP Claims, Evaluation Requirements

ASE_REQ.1 Security Target, IT Security Requirements, Evaluation Requirements

ASE_TSS.1 Security Target, TOE Summary Specification, Evaluation Requirements

Developer action elements:

ASE_INT.1.1D **The developer shall provide an ST introduction as part of the ST.**

Content and presentation of evidence elements:

ASE_INT.1.1C **The ST introduction shall contain an ST identification that provides the labelling and descriptive information necessary to control and identify the ST and the TOE to which it refers.**

ASE_INT.1.2C **The ST introduction shall contain an ST overview which summarises the ST in narrative form.**

ASE_INT.1.3C **The ST introduction shall contain a CC conformance claim that states any evaluatable claim of CC conformance for the TOE.**

Evaluator action elements:

ASE_INT.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ASE_INT.1.2E **The evaluator shall confirm that the ST introduction is coherent and internally consistent.**

ASE_INT.1.3E **The evaluator shall confirm that the ST introduction is consistent with the other parts of the ST.**

5.4 Security objectives (ASE_OBJ)

Objectives

The security objectives are a concise statement of the intended response to the security problem. Evaluation of the security objectives is required to demonstrate that the stated objectives adequately address the security problem. The security objectives are categorised as security objectives for the TOE and as security objectives for the environment. The security objectives for both the TOE and the environment must be shown to be traced back to the identified threats to be countered and/or policies and assumptions to be met by each.

ASE_OBJ.1 Security Target, Security Objectives, Evaluation Requirements

Dependencies:

ASE_ENV.1 Security Target, Security Environment, Evaluation Requirements

Developer action elements:

ASE_OBJ.1.1D **The developer shall provide a statement of security objectives as part of the ST.**

ASE_OBJ.1.2D **The developer shall provide the security objectives rationale.**

Content and presentation of evidence elements:

ASE_OBJ.1.1C **The statement of security objectives shall define the security objectives for the TOE and its environment.**

ASE_OBJ.1.2C **The security objectives for the TOE shall be clearly stated and traced back to the identified threats to be countered by the TOE and/or organisational security policies to be met by the TOE.**

ASE_OBJ.1.3C **The security objectives for the environment shall be clearly stated and traced back to aspects of identified threats not completely countered by the TOE and/or organisational security policies or assumptions not completely met by the TOE.**

ASE_OBJ.1.4C **The security objectives rationale shall demonstrate that the stated security objectives are suitable to counter the identified threats to security.**

ASE_OBJ.1.5C **The security objectives rationale shall demonstrate that the stated security objectives are suitable to cover all of the identified organisational security policies and assumptions.**

Evaluator action elements:

ASE_OBJ.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ASE_OBJ.1.2E **The evaluator shall confirm that the statement of security objectives is complete, coherent, and internally consistent.**

5.5 PP claims (ASE_PPC)

Objectives

The goal of the evaluation of the Security Target PP claims is to determine whether the ST is a correct instantiation of the PP.

Application notes

The family applies only in the case of a PP claim. In all other cases, no developer action and no evaluator action is necessary.

Although additional evaluation activity is necessary when a PP claim is made, the ST evaluation effort is generally smaller than in cases where no PP is used because it is possible to reuse the PP evaluation results for the ST evaluation.

ASE_PPC.1 Security Target, PP Claims, Evaluation Requirements

Dependencies:

ASE_OBJ.1 Security Target, Security Objectives, Evaluation Requirements

ASE_REQ.1 Security Target, IT Security Requirements, Evaluation Requirements

Developer action elements:

ASE_PPC.1.1D **The developer shall provide any PP claims as part of the ST.**

ASE_PPC.1.2D **The developer shall provide the PP claims rationale for each provided PP claim.**

Content and presentation of evidence elements:

ASE_PPC.1.1C **Each PP claim shall identify the PP for which compliance is being claimed, including qualifications needed for that claim.**

ASE_PPC.1.2C **Each PP claim shall identify the IT security requirements statements that satisfy the permitted operations of the PP or otherwise further qualify the PP requirements.**

ASE_PPC.1.3C **Each PP claim shall identify security objectives and IT security requirements statements contained in the ST that are in addition to those contained in the PP.**

Evaluator action elements:

ASE_PPC.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ASE_PPC.1.2E **The evaluator shall confirm that the PP claims are a correct instantiation of the PP.**

5.6 IT security requirements (ASE_REQ)

Objectives

The IT security requirements chosen for a TOE and presented or cited in an ST need to be evaluated in order to confirm that they are internally consistent and lead to the development of a TOE that will meet its security objectives.

This family presents evaluation requirements that permit the evaluator to determine that an ST is suitable for use as a statement of requirements for the corresponding TOE. The additional criteria necessary for the evaluation of explicitly stated requirements is covered in the ASE_SRE family.

Application notes

The term “IT security requirements” refers to “TOE security requirements” and the optionally included “security requirements for the IT environment”.

The term “TOE security requirements” refers to “TOE security functional requirements” and/or “TOE security assurance requirements.”

In the ASE_REQ.1 component, the word “appropriate” is used to indicate that certain elements allow options in certain cases. Which options are applicable depends on the given context in the ST. Detailed information for all these aspects is contained in Part 1, annex C.

ASE_REQ.1 Security Target, IT Security Requirements, Evaluation Requirements

Dependencies:

ASE_OBJ.1 Security Target, Security Objectives, Evaluation Requirements

Developer action elements:

ASE_REQ.1.1D **The developer shall provide a statement of IT security requirements as part of the ST.**

ASE_REQ.1.2D **The developer shall provide the security requirements rationale.**

Content and presentation of evidence elements:

ASE_REQ.1.1C **The statement of TOE security functional requirements shall identify the TOE security functional requirements drawn from CC Part 2 functional requirements components.**

ASE_REQ.1.2C **The statement of TOE security assurance requirements shall identify the TOE security assurance requirements drawn from CC Part 3 assurance requirements components.**

ASE_REQ.1.3C **The statement of TOE security assurance requirements should include a CC Evaluation Assurance Level (EAL) as defined in CC Part 3.**

ASE_REQ.1.4C **The evidence shall justify that the statement of TOE security assurance requirements is appropriate.**

ASE_REQ.1.5C **The ST shall, if appropriate, identify any security requirements for the IT environment.**

ASE_REQ.1.6C **Operations on IT security requirements included in the ST shall be identified and performed.**

ASE_REQ.1.7C **Dependencies among the IT security requirements included in the ST should be satisfied.**

ASE_REQ.1.8C **The evidence shall justify why any non-satisfaction of dependencies is appropriate.**

ASE_REQ.1.9C **The ST shall include a statement of the minimum strength of function level for the TOE security functional requirements, either SOF-basic, SOF-medium or SOF-high, as appropriate.**

ASE_REQ.1.10C **The ST shall identify any specific TOE security functional requirements for which an explicit strength of function is appropriate, together with the specific metric.**

ASE_REQ.1.11C **The security requirements rationale shall demonstrate that the minimum strength of function level for the ST together with any explicit strength of function claim is consistent with the security objectives for the TOE.**

ASE_REQ.1.12C **The security requirements rationale shall demonstrate that the IT security requirements are suitable to meet the security objectives.**

ASE_REQ.1.13C **The security requirements rationale shall demonstrate that the set of IT security requirements together forms a mutually supportive and internally consistent whole.**

Evaluator action elements:

ASE_REQ.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ASE_REQ.1.2E **The evaluator shall confirm that the statement of IT security requirements is complete, coherent, and internally consistent.**

5.7 Explicitly stated IT security requirements (ASE_SRE)

Objectives

If, after careful consideration, none of the Part 2 or Part 3 requirements components are readily applicable to all or parts of the IT security requirements, the ST author may state other requirements which do not reference the CC. The use of such requirements shall be justified.

This family presents evaluation requirements that permit the evaluator to determine that the explicitly stated requirements are clearly and unambiguously expressed. The evaluation of requirements taken from the CC in conjunction with valid explicitly stated security requirements is addressed by the ASE_REQ family.

Explicitly stated IT security requirements for a TOE presented or cited in an ST need to be evaluated in order to demonstrate that they are clearly and unambiguously expressed.

Application notes

Formulation of the explicitly stated requirements in a structure comparable to those of existing CC components and elements involves choosing similar labelling, manner of expression, and level of detail.

Using the CC requirements as a model means that the requirements can be clearly identified, that they are self-contained, and that the application of each requirement is feasible and will yield a meaningful evaluation result based on a compliance statement of the TOE for that particular requirement.

The term “IT security requirements” refers to “TOE security requirements” and the optionally included “security requirements for the IT environment”.

The term “TOE security requirements” refers to “TOE security functional requirements” and/or “TOE security assurance requirements.”

ASE_SRE.1 Security Target, Explicitly Stated IT Security Requirements, Evaluation Requirements

Dependencies:

ASE_REQ.1 Security Target, IT Security Requirements, Evaluation Requirements

Developer action elements:

ASE_SRE.1.1D **The developer shall provide a statement of IT security requirements as part of the ST.**

ASE_SRE.1.2D **The developer shall provide the security requirements rationale.**

Content and presentation of evidence elements:

ASE_SRE.1.1C **All TOE security requirements that are explicitly stated without reference to the CC shall be identified.**

ASE_SRE.1.2C **All security requirements for the IT environment that are explicitly stated without reference to the CC shall be identified.**

ASE_SRE.1.3C **The evidence shall justify why the security requirements had to be explicitly stated.**

ASE_SRE.1.4C **The explicitly stated IT security requirements shall use the CC requirements components, families and classes as a model for presentation.**

ASE_SRE.1.5C **The explicitly stated IT security requirements shall be measurable and state objective evaluation requirements such that compliance or noncompliance of a TOE can be determined and systematically demonstrated.**

ASE_SRE.1.6C **The explicitly stated IT security requirements shall be clearly and unambiguously expressed.**

ASE_SRE.1.7C **The security requirements rationale shall demonstrate that the assurance requirements are applicable and appropriate to support any explicitly stated TOE security functional requirements.**

Evaluator action elements:

ASE_SRE.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ASE_SRE.1.2E **The evaluator shall determine that all of the dependencies of the explicitly stated IT security requirements have been identified.**

5.8 TOE summary specification (ASE_TSS)

Objectives

The TOE summary specification provides a high-level definition of the security functions claimed to meet the functional requirements and of the assurance measures taken to meet the assurance requirements.

Application notes

The relationship between the IT security functions and the TOE security functional requirements can be a “many to many” relationship. Nevertheless, every security function shall contribute to the satisfaction of at least one security requirement in order be able to clearly define the TSF. Security functions that do not fulfil this requirement should normally not be necessary. Note, however, that the requirement that a security function contributes to the satisfaction of at least one security requirement is worded in a quite general manner, so that all the security functions found to be useful for the TOE should be justifiable.

The statement of assurance measures is of specific relevance in all those cases where assurance requirements not taken from the CC are included in the ST. If the TOE security assurance requirements in the ST are exclusively based on CC evaluation assurance levels or other CC assurance components, then the assurance measures could be presented in the form of a reference to the documents that show that the assurance requirements are met.

In the ASE_TSS.1 component, the word “appropriate” is used to indicate that certain elements allow options in certain cases. Which options are applicable depends on the given context in the ST. Detailed information for all these aspects is contained in Part 1, annex C.

ASE_TSS.1 Security Target, TOE Summary Specification, Evaluation Requirements

Dependencies:

ASE_REQ.1 Security Target, IT Security Requirements, Evaluation Requirements

Developer action elements:

ASE_TSS.1.1D **The developer shall provide a TOE summary specification as part of the ST.**

ASE_TSS.1.2D **The developer shall provide the TOE summary specification rationale.**

Content and presentation of evidence elements:

ASE_TSS.1.1C **The TOE summary specification shall describe the IT security functions and the assurance measures of the TOE.**

ASE_TSS.1.2C **The TOE summary specification shall trace the IT security functions to the TOE security functional requirements such that it can be seen which IT security functions satisfy which TOE security functional requirements and that every IT security function contributes to the satisfaction of at least one TOE security functional requirement.**

ASE_TSS.1.3C **The IT security functions shall be defined in an informal style to a level of detail necessary for understanding their intent.**

ASE_TSS.1.4C **All references to security mechanisms included in the ST shall be traced to the relevant security functions so that it can be seen which security mechanisms are used in the implementation of each function.**

ASE_TSS.1.5C **The TOE summary specification rationale shall demonstrate that the IT security functions are suitable to meet TOE security functional requirements.**

ASE_TSS.1.6C **The TOE summary specification rationale shall demonstrate that the combination of the specified IT security functions work together so as to satisfy the TOE security functional requirements.**

ASE_TSS.1.7C **The TOE summary specification shall trace the assurance measures to the assurance requirements so that it can be seen which measures contribute to the satisfaction of which requirements.**

ASE_TSS.1.8C **The TOE summary specification rationale shall demonstrate that the assurance measures meet all assurance requirements of the TOE.**

ASE_TSS.1.9C **The TOE summary specification shall identify all IT security functions that are realised by a probabilistic or permutational mechanism, as appropriate.**

ASE_TSS.1.10C **The TOE summary specification shall, for each IT security function for which it is appropriate, state the strength of function claim either as a specific metric, or as SOF-basic, SOF-medium or SOF-high.**

Evaluator action elements:

ASE_TSS.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ASE_TSS.1.2E **The evaluator shall confirm that the TOE summary specification is complete, coherent, and internally consistent.**

6 Evaluation assurance levels

The Evaluation Assurance Levels (EALs) provide an increasing scale that balances the level of assurance obtained with the cost and feasibility of acquiring that degree of assurance. The CC approach identifies the separate concepts of assurance in a TOE at the end of the evaluation, and of maintenance of that assurance during the operational use of the TOE.

It is important to note that not all families and components from Part 3 are included in the EALs. This is not to say that these do not provide meaningful and desirable assurances. Instead, it is expected that these families and components will be considered for augmentation of an EAL in those PPs and STs for which they provide utility.

6.1 Evaluation assurance level (EAL) overview

Table 6.1 represents a summary of the EALs. The columns represent a hierarchically ordered set of EALs, while the rows represent assurance families. Each point in the resulting matrix identifies a specific assurance component where applicable.

As outlined in the next section, seven hierarchically ordered evaluation assurance levels are defined in the CC for the rating of a TOE's assurance. They are hierarchically ordered inasmuch as each EAL represents more assurance than all lower EALs. The increase in assurance from EAL to EAL is accomplished by *substitution* of a hierarchically higher assurance component from the same assurance family (i.e. increasing rigour, scope, and/or depth) and from the *addition* of assurance components from other assurance families (i.e. adding new requirements).

These EALs consist of an appropriate combination of assurance components as described in Chapter 2 of this Part. More precisely, each EAL includes no more than one component of each assurance family and all assurance dependencies of every component are addressed.

While the EALs are defined in the CC, it is possible to represent other combinations of assurance. Specifically, the notion of “augmentation” allows the addition of assurance components (from assurance families not already included in the EAL) or the substitution of assurance components (with another hierarchically higher assurance component in the same assurance family) to an EAL. Of the assurance constructs defined in the CC, only EALs may be augmented. The notion of an “EAL minus a constituent assurance component” is not recognised by the CC as a valid claim. Augmentation carries with it the obligation on the part of the claimant to justify the utility and added value of the added assurance component to the EAL. An EAL may also be extended with explicitly stated assurance requirements.

6.2 Evaluation assurance level details

The following sections provide definitions of the EALs, highlighting differences between the specific requirements and the prose characterisations of those requirements using bold type.

Assurance Class	Assurance Family	Assurance Components by Evaluation Assurance Level						
		EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
Configuration management	ACM_AUT				**1**	1	**2**	2
	ACM_CAP	**1**	**2**	**3**	**4**	4	**5**	5
	ACM_SCP			**1**	**2**	**3**	3	3
Delivery and operation	ADO_DEL		**1**	1	**2**	2	2	**3**
	ADO_IGS	**1**	1	1	1	1	1	1
Development	ADV_FSP	**1**	1	1	**2**	**3**	3	**4**
	ADV_HLD		**1**	**2**	2	**3**	**4**	**5**
	ADV_IMP				**1**	**2**	**3**	3
	ADV_INT					**1**	**2**	**3**
	ADV_LLD				**1**	1	**2**	2
	ADV_RCR	**1**	1	1	1	**2**	2	**3**
	ADV_SPM				**1**	**3**	3	3
Guidance documents	AGD_ADM	**1**	1	1	1	1	1	1
	AGD_USR	**1**	1	1	1	1	1	1
Life cycle support	ALC_DVS			**1**	1	1	**2**	2
	ALC_FLR							
	ALC_LCD				**1**	**2**	2	**3**
	ALC_TAT				**1**	**2**	**3**	3
Tests	ATE_COV		**1**	**2**	2	2	**3**	3
	ATE_DPT			**1**	1	**2**	2	**3**
	ATE_FUN		**1**	1	1	1	**2**	2
	ATE_IND	**1**	**2**	2	2	2	2	**3**
Vulnerability assessment	AVA_CCA					**1**	**2**	2
	AVA_MSU			**1**	**2**	2	**3**	3
	AVA_SOF		**1**	1	1	1	1	1
	AVA_VLA		**1**	1	**2**	**3**	**4**	4

Table 6.1 - Evaluation assurance level summary

6.2.1 Evaluation assurance level 1 (EAL1) - functionally tested

Objectives

EAL1 is applicable where some confidence in correct operation is required, but the threats to security are not viewed as serious. It will be of value where independent assurance is required to support the contention that due care has been exercised with respect to the protection of personal or similar information.

EAL1 provides an evaluation of the TOE as made available to the customer, including independent testing against a specification, and an examination of the guidance documentation provided. It is intended that an EAL1 evaluation could be successfully conducted without assistance from the developer of the TOE, and for minimal outlay.

An evaluation at this level should provide evidence that the TOE functions in a manner consistent with its documentation, and that it provides useful protection against identified threats.

Assurance components

EAL1 (see Table 6.2) provides a basic level of assurance by an analysis of the security functions using a functional and interface specification and guidance documentation, to understand the security behaviour.

The analysis is supported by independent testing of the TOE security functions.

This EAL provides a meaningful increase in assurance over an unevaluated IT product or system (TOE).

Assurance class	Assurance components
Configuration management	ACM_CAP.1 Version numbers
Delivery and operation	ADO_IGS.1 Installation, generation, and start-up procedures
Development	ADV_FSP.1 Informal functional specification
	ADV_RCR.1 Informal correspondence demonstration
Guidance documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Tests	ATE_IND.1 Independent testing - conformance

Table 6.2 - EAL1

6.2.2 Evaluation assurance level 2 (EAL2) - structurally tested

Objectives

EAL2 requires the co-operation of the developer in terms of the delivery of design information and test results, but should not demand more effort on the part of the developer than is consistent with good commercial practice. As such it should not require a substantially increased investment of cost or time.

EAL2 is therefore applicable in those circumstances where developers or users require a low to moderate level of independently assured security in the absence of ready availability of the complete development record. Such a situation may arise when securing legacy systems, or where access to the developer may be limited.

Assurance components

EAL2 (see Table **6.3**) provides assurance by an analysis of the security functions, using a functional and interface specification, guidance documentation **and the high-level design** of the TOE, to understand the security behaviour**.**

The analysis is supported by independent testing of the TOE security functions**, evidence of developer testing based on the functional specification, selective independent confirmation of the developer test results, strength of function analysis and evidence of a developer search for obvious vulnerabilities (e.g. those in the public domain).**

EAL2 also provides assurance through a configuration list for the TOE, and evidence of secure delivery procedures.

This EAL represents a meaningful increase in assurance from EAL1 by requiring developer testing, a vulnerability analysis, and independent testing based upon more detailed TOE specifications.

Assurance class	Assurance components
Configuration management	**ACM_CAP.2 Configuration items**
Delivery and operation	**ADO_DEL.1 Delivery procedures**
	ADO_IGS.1 Installation, generation, and start-up procedures
Development	ADV_FSP.1 Informal functional specification
	ADV_HLD.1 Descriptive high-level design
	ADV_RCR.1 Informal correspondence demonstration
Guidance documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Tests	**ATE_COV.1 Evidence of coverage**
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing - sample
Vulnerability assessment	**AVA_SOF.1 Strength of TOE security function evaluation**
	AVA_VLA.1 Developer vulnerability analysis

Table 6.3 - EAL2

6.2.3 Evaluation assurance level 3 (EAL3) - methodically tested and checked

Objectives

EAL3 permits a conscientious developer to gain maximum assurance from positive security engineering at the design stage without substantial alteration of existing sound development practices.

EAL3 is applicable in those circumstances where developers or users require a moderate level of independently assured security, and require a thorough investigation of the TOE and its development without substantial re-engineering.

Assurance components

EAL3 (see Table **6.4**) provides assurance by an analysis of the security functions, using a functional and interface specification, guidance documentation, and the high-level design of the TOE, to understand the security behaviour**.**

The analysis is supported by independent testing of the TOE security functions, evidence of developer testing based on the functional specification **and high-level design**, selective independent confirmation of the developer test results, strength of function analysis, evidence of a developer search for obvious vulnerabilities (e.g. those in the public domain).

EAL3 also provides assurance through **the use of development environment controls, TOE configuration management,** and evidence of secure delivery procedures**.**

This EAL represents a meaningful increase in assurance from EAL2 by requiring more complete testing coverage of the security functions and mechanisms and/or procedures that provide some confidence that the TOE will not be tampered with during development.

Assurance class	Assurance components
Configuration management	**ACM_CAP.3 Authorisation controls**
	ACM_SCP.1 TOE CM coverage
Delivery and operation	ADO_DEL.1 Delivery procedures
	ADO_IGS.1 Installation, generation, and start-up procedures
Development	ADV_FSP.1 Informal functional specification
	ADV_HLD.2 Security enforcing high-level design
	ADV_RCR.1 Informal correspondence demonstration
Guidance documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Life cycle support	**ALC_DVS.1 Identification of security measures**
Tests	**ATE_COV.2 Analysis of coverage**
	ATE_DPT.1 Testing - high level design
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing - sample
Vulnerability assessment	**AVA_MSU.1 Examination of guidance**
	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.1 Developer vulnerability analysis

Table 6.4 - EAL3

6.2.4 Evaluation assurance level 4 (EAL4) - methodically designed, tested, and reviewed

Objectives

EAL4 permits a developer to gain maximum assurance from positive security engineering based on good commercial development practices which, though rigorous, do not require substantial specialist knowledge, skills, and other resources. EAL4 is the highest level at which it is likely to be economically feasible to retrofit to an existing product line.

EAL4 is therefore applicable in those circumstances where developers or users require a moderate to high level of independently assured security in conventional commodity TOEs and are prepared to incur additional security specific engineering costs.

Assurance components

EAL4 (see Table **6.5**) provides assurance by an analysis of the security functions, using a functional and **complete** interface specification**,** guidance documentation, the high-level **and low-level** design of the TOE**, and a subset of the implementation**, to understand the security behaviour**. Assurance is additionally gained through an informal model of the TOE security policy.**

The analysis is supported by independent testing of the TOE security functions, evidence of developer testing based on the functional specification and high-level design, selective independent confirmation of the developer test results, strength of function analysis, evidence of a developer search for vulnerabilities, **and an independent vulnerability analysis demonstrating resistance to penetration attackers with a low attack potential**.

EAL4 also provides assurance through the use of development environment controls and **additional** TOE configuration management **including automation,** and evidence of secure delivery procedures.

This EAL represents a meaningful increase in assurance from EAL3 by requiring more design description, a subset of the implementation, and improved mechanisms and/or procedures that provide confidence that the TOE will not be tampered with during development or delivery.

Assurance class	Assurance components
Configuration management	**ACM_AUT.1 Partial CM automation**
	ACM_CAP.4 Generation support and acceptance procedures
	ACM_SCP.2 Problem tracking CM coverage
Delivery and operation	**ADO_DEL.2 Detection of modification**
	ADO_IGS.1 Installation, generation, and start-up procedures
Development	**ADV_FSP.2 Fully defined external interfaces**
	ADV_HLD.2 Security enforcing high-level design
	ADV_IMP.1 Subset of the implementation of the TSF
	ADV_LLD.1 Descriptive low-level design
	ADV_RCR.1 Informal correspondence demonstration
	ADV_SPM.1 Informal TOE security policy model
Guidance documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Life cycle support	**ALC_DVS.1 Identification of security measures**
	ALC_LCD.1 Developer defined life-cycle model
	ALC_TAT.1 Well-defined development tools
Tests	ATE_COV.2 Analysis of coverage
	ATE_DPT.1 Testing - high level design
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing - sample
Vulnerability assessment	**AVA_MSU.2 Validation of analysis**
	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.2 Independent vulnerability analysis

Table 6.5 - EAL4

6.2.5 Evaluation assurance level 5 (EAL5) - semiformally designed and tested

Objectives

EAL5 permits a developer to gain maximum assurance from security engineering based upon rigorous commercial development practices supported by moderate application of specialist security engineering techniques. Such a TOE will probably be designed and developed with the intent of achieving EAL5 assurance. It is likely that the additional costs attributable to the EAL5 requirements relative to rigorous development without the application of specialised techniques will not be large.

EAL5 is therefore applicable in those circumstances where developers or users require a high level of independently assured security in a planned development and require a rigorous development approach without incurring unreasonable costs attributable to specialist security engineering techniques.

Assurance components

EAL5 (see Table **6.6**) provides assurance by an analysis of the security functions, using a functional and complete interface specification**,** guidance documentation, the high-level and low-level design of the TOE, and **all** of the implementation, to understand the security behaviour**.** Assurance is additionally gained through **a formal** model of the TOE security policy **and a semiformal presentation of the functional specification and high-level design and a semiformal demonstration of correspondence between them. A modular TOE design is also required.**

The analysis is supported by independent testing of the TOE security functions, evidence of developer testing based on the functional specification, high-level design **and low-level design**, selective independent confirmation of the developer test results, strength of function analysis, evidence of a developer search for vulnerabilities, and an independent vulnerability analysis demonstrating resistance to penetration attackers with a **moderate** attack potential. **The analysis also includes validation of the developer's covert channel analysis.**

EAL5 also provides assurance through the use of a development environment controls, and **comprehensive** TOE configuration management including automation, and evidence of secure delivery procedures.

This EAL represents a meaningful increase in assurance from EAL4 by requiring semiformal design descriptions, the entire implementation, a more structured (and hence analysable) architecture, covert channel analysis, and improved mechanisms and/or procedures that provide confidence that the TOE will not be tampered with during development.

Assurance class	Assurance components
Configuration management	ACM_AUT.1 Partial CM automation
	ACM_CAP.4 Generation support and acceptance procedures
	ACM_SCP.3 Development tools CM coverage
Delivery and operation	ADO_DEL.2 Detection of modification
	ADO_IGS.1 Installation, generation, and start-up procedures
Development	**ADV_FSP.3 Semiformal functional specification**
	ADV_HLD.3 Semiformal high-level design
	ADV_IMP.2 Implementation of the TSF
	ADV_INT.1 Modularity
	ADV_LLD.1 Descriptive low-level design
	ADV_RCR.2 Semiformal correspondence demonstration
	ADV_SPM.3 Formal TOE security policy model
Guidance documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Life cycle support	ALC_DVS.1 Identification of security measures
	ALC_LCD.2 Standardised life-cycle model
	ALC_TAT.2 Compliance with implementation standards
Tests	ATE_COV.2 Analysis of coverage
	ATE_DPT.2 Testing - low level design
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing - sample
Vulnerability assessment	**AVA_CCA.1 Covert channel analysis**
	AVA_MSU.2 Validation of analysis
	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.3 Moderately resistant

Table 6.6 - EAL5

6.2.6 Evaluation assurance level 6 (EAL6) - semiformally verified design and tested

Objectives

EAL6 permits developers to gain high assurance from application of security engineering techniques to a rigorous development environment in order to produce a premium TOE for protecting high value assets against significant risks.

EAL6 is therefore applicable to the development of security TOEs for application in high risk situations where the value of the protected assets justifies the additional costs.

Assurance components

EAL6 (see Table **6.7**) provides assurance by an analysis of the security functions, using a functional and complete interface specification**,** guidance documentation, the high-level and low-level design of the of the TOE, and **a structured presentation** of the implementation, to understand the security behaviour**.** Assurance is additionally gained through a formal model of the TOE security policy, a semiformal presentation of the functional specification, high-level design**, and low-level design** and a semiformal demonstration of correspondence between them. A modular **and layered** TOE design is also required.

The analysis is supported by independent testing of the TOE security functions, evidence of developer testing based on the functional specification, high-level design and low-level design, selective independent confirmation of the developer test results, strength of function analysis, evidence of a developer search for vulnerabilities, and an independent vulnerability analysis demonstrating resistance to penetration attackers with a **high** attack potential. The analysis also includes validation of the developer’s **systematic** covert channel analysis.

EAL6 also provides assurance through the use of a **structured development process,** development environment controls, and comprehensive TOE configuration management including **complete** automation, and evidence of secure delivery procedures.

This EAL represents a meaningful increase in assurance from EAL5 by requiring more comprehensive analysis, a structured representation of the implementation, more architectural structure (e.g. layering), more comprehensive independent vulnerability analysis, systematic covert channel identification, and improved configuration management and development environment controls.

Assurance class	Assurance components
Configuration management	**ACM_AUT.2 Complete CM automation**
	ACM_CAP.5 Advanced support
	ACM_SCP.3 Development tools CM coverage
Delivery and operation	ADO_DEL.2 Detection of modification
	ADO_IGS.1 Installation, generation, and start-up procedures
Development	**ADV_FSP.3 Semiformal functional specification**
	ADV_HLD.4 Semiformal high-level explanation
	ADV_IMP.3 Structured implementation of the TSF
	ADV_INT.2 Reduction of complexity
	ADV_LLD.2 Semiformal low-level design
	ADV_RCR.2 Semiformal correspondence demonstration
	ADV_SPM.3 Formal TOE security policy model
Guidance documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Life cycle support	**ALC_DVS.2 Sufficiency of security measures**
	ALC_LCD.2 Standardised life-cycle model
	ALC_TAT.3 Compliance with implementation standards - all parts
Tests	**ATE_COV.3 Rigorous analysis of coverage**
	ATE_DPT.2 Testing - low level design
	ATE_FUN.2 Ordered functional testing
	ATE_IND.2 Independent testing - sample
Vulnerability assessment	**AVA_CCA.2 Systematic covert channel analysis**
	AVA_MSU.3 Analysis and testing for insecure states
	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.4 Highly resistant

Table 6.7 - EAL6

6.2.7 Evaluation assurance level 7 (EAL7) - formally verified design and tested

Objectives

EAL7 is applicable to the development of security TOEs for application in extremely high risk situations and/or where the high value of the assets justifies the higher costs. Practical application of EAL7 is currently limited to TOEs with tightly focused security functionality that is amenable to extensive formal analysis.

Assurance components

EAL7 (see Table **6.8**) provides assurance by an analysis of the security functions, using a functional and complete interface specification, guidance documentation, the high-level and low-level design of the TOE, and a structured presentation of the implementation, to understand the security behaviour. Assurance is additionally gained through a formal model of the TOE security policy**, a formal presentation of the functional specification and high-level design**, a semiformal presentation of the low-level design, and **formal and** semiformal demonstration of correspondence between them**, as appropriate**. A modular, layered **and simple** TOE design is also required.

The analysis is supported by independent testing of the TOE security functions, evidence of developer testing based on the functional specification high-level design, low-level design **and implementation representation**, **complete** independent confirmation of the developer test results, strength of function analysis, evidence of a developer search for vulnerabilities, and an independent vulnerability analysis demonstrating resistance to penetration attackers with a **high** attack potential. The analysis also includes validation of the developer's systematic covert channel analysis.

EAL7 also provides assurance through the use of a structured development process, development environment controls, and comprehensive TOE configuration management including complete automation, and evidence of secure delivery procedures.

This EAL represents a meaningful increase in assurance from EAL6 by requiring more comprehensive analysis using formal representations and formal correspondence, and comprehensive testing.

Assurance class	Assurance components
Configuration management	ACM_AUT.2 Complete CM automation
	ACM_CAP.5 Advanced support
	ACM_SCP.3 Development tools CM coverage
Delivery and operation	**ADO_DEL.3 Prevention of modification**
	ADO_IGS.1 Installation, generation, and start-up procedures
Development	**ADV_FSP.4 Formal functional specification**
	ADV_HLD.5 Formal high-level design
	ADV_IMP.3 Structured implementation of the TSF
	ADV_INT.3 Minimisation of complexity
	ADV_LLD.2 Semiformal low-level design
	ADV_RCR.3 Formal correspondence demonstration
	ADV_SPM.3 Formal TOE security policy model
Guidance documents	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Life cycle support	ALC_DVS.2 Sufficiency of security measures
	ALC_LCD.3 Measurable life-cycle model
	ALC_TAT.3 Compliance with implementation standards - all parts
Tests	ATE_COV.3 Rigorous analysis of coverage
	ATE_DPT.3 Testing - implementation
	ATE_FUN.2 Ordered functional testing
	ATE_IND.3 Independent testing - complete
Vulnerability assessment	AVA_CCA.2 Systematic covert channel analysis
	AVA_MSU.3 Analysis and testing for insecure states
	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.4 Highly resistant

Table 6.8 - EAL7

7 Assurance classes, families, and components

The next seven chapters provide the detailed requirements, presented in alphabetical order, of each of the assurance components, grouped by class and family.

8 Class ACM: Configuration management

Configuration management (CM) is one method or means for establishing that the functional requirements and specifications are realised in the implementation of the TOE. CM meets these objectives by requiring discipline and control in the processes of refinement and modification of the TOE and the related information. CM systems are put in place to ensure the integrity of the portions of the TOE that they control, by providing a method of tracking any changes, and by ensuring that all changes are authorised.

Figure 8.1 shows the families within this class, and the hierarchy of components within the families.

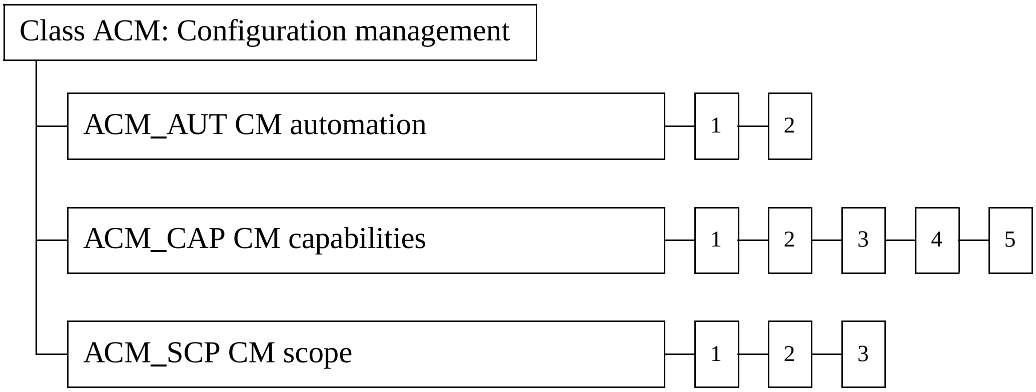

Figure 8.1 -Configuration management class decomposition

8.1 CM automation (ACM_AUT)

Objectives

The objective of introducing automated CM tools is to increase the effectiveness of the CM system. While both automated and manual CM systems can be bypassed, ignored, or prove insufficient to prevent unauthorised modification, automated systems are less susceptible to human error or negligence.

Component levelling

The components in this family are levelled on the basis of the set of configuration items that are controlled through automated means.

Application notes

ACM_AUT.1.1C introduces a requirement that is related to the implementation representation of the TOE. The implementation representation of the TOE consists of all hardware, software, and firmware that comprise the physical TOE. In the case of a software-only TOE, the implementation representation may consist solely of source and object code.

ACM_AUT.1.2C introduces a requirement that the CM system provide an automated means to support the generation of the TOE. This requires that the CM system provide an automated means to assist in determining that the correct configuration items are used in generating the TOE.

ACM_AUT.2.5C introduces a requirement that the CM system provide an automated means to ascertain the changes between the TOE and its preceding version. If no previous version of the TOE exists, the developer still needs to provide an automated means to ascertain the changes between the TOE and a future version of the TOE.

ACM_AUT.1 Partial CM automation

Objectives

In development environments where the implementation representation is complex or is being developed by multiple developers, it is difficult to control changes without the support of automated tools. In particular, these automated tools need to be able to support the numerous changes that occur during development and ensure that those changes are authorised. It is the objective of this component to ensure that the implementation representation is controlled through automated means.

Dependencies:

ACM_CAP.3 Authorisation controls

Developer action elements:

ACM_AUT.1.1D **The developer shall use a CM system.**

ACM_AUT.1.2D **The developer shall provide a CM plan.**

Content and presentation of evidence elements:

ACM_AUT.1.1C **The CM system shall provide an automated means by which only authorised changes are made to the TOE implementation representation.**

ACM_AUT.1.2C **The CM system shall provide an automated means to support the generation of the TOE.**

ACM_AUT.1.3C **The CM plan shall describe the automated tools used in the CM system.**

ACM_AUT.1.4C **The CM plan shall describe how the automated tools are used in the CM system.**

Evaluator action elements:

ACM_AUT.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ACM_AUT.2 Complete CM automation

Objectives

In development environments where the configuration items are complex or are being developed by multiple developers, it is difficult to control changes without the support of automated tools. In particular, these automated tools need to be able to support the numerous changes that occur during development and ensure that those changes are authorised. It is the objective of this component to ensure that all configuration items are controlled through automated means.

Providing an automated means of ascertaining changes between versions of the TOE and identifying which configuration items are affected by modifications to other configuration items assists in determining the impact of the changes between successive versions of the TOE. This in turn can provide valuable information in determining whether changes to the TOE result in all configuration items being consistent with one another.

Dependencies:

ACM_CAP.3 Authorisation controls

Developer action elements:

ACM_AUT.2.1D The developer shall use a CM system.

ACM_AUT.2.2D The developer shall provide a CM plan.

Content and presentation of evidence elements:

ACM_AUT.2.1C The CM system shall provide an automated means by which only authorised changes are made to the TOE implementation representation**, and to all other configuration items.**

ACM_AUT.2.2C The CM system shall provide an automated means to support the generation of the TOE.

ACM_AUT.2.3C The CM plan shall describe the automated tools used in the CM system.

ACM_AUT.2.4C The CM plan shall describe how the automated tools are used in the CM system.

ACM_AUT.2.5C **The CM system shall provide an automated means to ascertain the changes between the TOE and its preceding version.**

ACM_AUT.2.6C **The CM system shall provide an automated means to identify all other configuration items that are affected by the modification of a given configuration item.**

Evaluator action elements:

ACM_AUT.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

8.2 CM capabilities (ACM_CAP)

Objectives

The capabilities of the CM system address the likelihood that accidental or unauthorised modifications of the configuration items will occur. The CM system should ensure the integrity of the TOE from the early design stages through all subsequent maintenance efforts.

The objectives of this family include the following:

a) ensuring that the TOE is correct and complete before it is sent to the consumer;

b) ensuring that no configuration items are missed during evaluation;

c) preventing unauthorised modification, addition, or deletion of TOE configuration items.

Component levelling

The components in this family are levelled on the basis of the CM system capabilities, the scope of the CM documentation provided by the developer, and whether the developer provides justification that the CM system meets its security requirements.

Application notes

ACM_CAP.2 introduces several elements which refer to configuration items. The ACM_SCP family contains requirements for the configuration items to be tracked by the CM system.

ACM_CAP.2.3C introduces a requirement that a configuration list be provided. The configuration list contains all configuration items that are maintained by the CM system.

ACM_CAP.2.6C introduces a requirement that the CM system uniquely identify all configuration items. This also requires that modifications to configuration items result in a new, unique identifier being assigned.

ACM_CAP.3.8C introduces the requirement that the evidence shall demonstrate that the CM system operates in accordance with the CM plan. Examples of such evidence might be documentation such as screen snapshots or audit trail output from the CM system, or a detailed demonstration of the CM system by the developer. The evaluator is responsible for determining that this evidence is sufficient to show that the CM system operates in accordance with the CM plan.

ACM_CAP.3.9C introduces the requirement that evidence be provided to show that all configuration items are being maintained under the CM system. Since a configuration item refers to an item that is on the configuration list, this requirement states that all items on the configuration list are maintained under the CM system.

ACM_CAP.4.11C introduces the requirement that the CM system support the generation of the TOE. This requires that the CM system provide information and/or electronic means to assist in determining that the correct configuration items are used in generating the TOE.

ACM_CAP.1 Version numbers

Objectives

A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be aware of which instance of the TOE they are using.

Dependencies:

No dependencies.

Developer action elements:

ACM_CAP.1.1D **The developer shall provide a reference for the TOE.**

Content and presentation of evidence elements:

ACM_CAP.1.1C **The reference for the TOE shall be unique to each version of the TOE.**

ACM_CAP.1.2C **The TOE shall be labelled with its reference.**

Evaluator action elements:

ACM_CAP.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ACM_CAP.2 Configuration items

Objectives

A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated.

Unique identification of the configuration items leads to a clearer understanding of the composition of the TOE, which in turn helps to determine those items which are subject to the evaluation requirements for the TOE.

Dependencies:

No dependencies.

Developer action elements:

ACM_CAP.2.1D The developer shall provide a reference for the TOE.

ACM_CAP.2.2D **The developer shall use a CM system.**

ACM_CAP.2.3D **The developer shall provide CM documentation.**

Content and presentation of evidence elements:

ACM_CAP.2.1C The reference for the TOE shall be unique to each version of the TOE.

ACM_CAP.2.2C The TOE shall be labelled with its reference.

ACM_CAP.2.3C **The CM documentation shall include a configuration list.**

ACM_CAP.2.4C **The configuration list shall describe the configuration items that comprise the TOE.**

ACM_CAP.2.5C **The CM documentation shall describe the method used to uniquely identify the configuration items.**

ACM_CAP.2.6C **The CM system shall uniquely identify all configuration items.**

Evaluator action elements:

ACM_CAP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_CAP.3 Authorisation controls

Objectives

A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated.

Unique identification of the configuration items leads to a clearer understanding of the composition of the TOE, which in turn helps to determine those items which are subject to the evaluation requirements for the TOE.

Providing controls to ensure that unauthorised modifications are not made to the TOE, and ensuring proper functionality and use of the CM system, helps to maintain the integrity of the TOE.

Dependencies:

ACM_SCP.1 TOE CM coverage

ALC_DVS.1 Identification of security measures

Developer action elements:

ACM_CAP.3.1D The developer shall provide a reference for the TOE.

ACM_CAP.3.2D The developer shall use a CM system.

ACM_CAP.3.3D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_CAP.3.1C The reference for the TOE shall be unique to each version of the TOE.

ACM_CAP.3.2C The TOE shall be labelled with its reference.

ACM_CAP.3.3C The CM documentation shall include a configuration list **and a CM plan**.

ACM_CAP.3.4C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.3.5C The CM documentation shall describe the method used to uniquely identify the configuration items.

ACM_CAP.3.6C The CM system shall uniquely identify all configuration items.

ACM_CAP.3.7C **The CM plan shall describe how the CM system is used.**

ACM_CAP.3.8C **The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.**

ACM_CAP.3.9C **The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.**

ACM_CAP.3.10C **The CM system shall provide measures such that only authorised changes are made to the configuration items.**

Evaluator action elements:

ACM_CAP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_CAP.4 Generation support and acceptance procedures

Objectives

A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated.

Unique identification of the configuration items leads to a clearer understanding of the composition of the TOE, which in turn helps to determine those items which are subject to the evaluation requirements for the TOE.

Providing controls to ensure that unauthorised modifications are not made to the TOE, and ensuring proper functionality and use of the CM system, helps to maintain the integrity of the TOE.

The purpose of acceptance procedures is to confirm that any creation or modification of configuration items is authorised.

Dependencies:

ACM_SCP.1 TOE CM coverage

ALC_DVS.1 Identification of security measures

Developer action elements:

ACM_CAP.4.1D The developer shall provide a reference for the TOE.

ACM_CAP.4.2D The developer shall use a CM system.

ACM_CAP.4.3D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_CAP.4.1C The reference for the TOE shall be unique to each version of the TOE.

ACM_CAP.4.2C The TOE shall be labelled with its reference.

ACM_CAP.4.3C The CM documentation shall include a configuration list**,** a CM plan**, and an acceptance plan.**

ACM_CAP.4.4C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.4.5C The CM documentation shall describe the method used to uniquely identify the configuration items.

ACM_CAP.4.6C The CM system shall uniquely identify all configuration items.

ACM_CAP.4.7C The CM plan shall describe how the CM system is used.

ACM_CAP.4.8C The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

ACM_CAP.4.9C The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_CAP.4.10C The CM system shall provide measures such that only authorised changes are made to the configuration items.

ACM_CAP.4.11C **The CM system shall support the generation of the TOE.**

ACM_CAP.4.12C **The acceptance plan shall describe the procedures used to accept modified or newly created configuration items as part of the TOE.**

Evaluator action elements:

ACM_CAP.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_CAP.5 Advanced support

Objectives

A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated.

Unique identification of the configuration items leads to a clearer understanding of the composition of the TOE, which in turn helps to determine those items which are subject to the evaluation requirements for the TOE.

Providing controls to ensure that unauthorised modifications are not made to the TOE, and ensuring proper functionality and use of the CM system, helps to maintain the integrity of the TOE.

The purpose of acceptance procedures is to confirm that any creation or modification of configuration items is authorised.

Integration procedures help to ensure that generation of the TOE from a managed set of configuration items is correctly performed in an authorised manner.

Requiring that the CM system be able to identify the master copy of the material used to generate the TOE helps to ensure that the integrity of this material is preserved by the appropriate technical, physical and procedural safeguards.

Dependencies:

ACM_SCP.1 TOE CM coverage

ALC_DVS.2 Sufficiency of security measures

Developer action elements:

ACM_CAP.5.1D The developer shall provide a reference for the TOE.

ACM_CAP.5.2D The developer shall use a CM system.

ACM_CAP.5.3D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_CAP.5.1C The reference for the TOE shall be unique to each version of the TOE.

ACM_CAP.5.2C The TOE shall be labelled with its reference.

ACM_CAP.5.3C The CM documentation shall include a configuration list, a CM plan, an acceptance plan**, and integration procedures.**

ACM_CAP.5.4C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.5.5C The CM documentation shall describe the method used to uniquely identify the configuration items.

ACM_CAP.5.6C The CM system shall uniquely identify all configuration items.

ACM_CAP.5.7C The CM plan shall describe how the CM system is used.

ACM_CAP.5.8C The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

ACM_CAP.5.9C The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_CAP.5.10C The CM system shall provide measures such that only authorised changes are made to the configuration items.

ACM_CAP.5.11C The CM system shall support the generation of the TOE.

ACM_CAP.5.12C The acceptance plan shall describe the procedures used to accept modified or newly created configuration items as part of the TOE.

ACM_CAP.5.13C **The integration procedures shall describe how the CM system is applied in the TOE manufacturing process.**

ACM_CAP.5.14C **The CM system shall require that the person responsible for accepting a configuration item into CM is not the person who developed it.**

ACM_CAP.5.15C **The CM system shall clearly identify the configuration items that comprise the TSF.**

ACM_CAP.5.16C **The CM system shall support the audit of all modifications to the TOE, including as a minimum the originator, date, and time in the audit trail.**

ACM_CAP.5.17C **The CM system shall be able to identify the master copy of all material used to generate the TOE.**

ACM_CAP.5.18C **The CM documentation shall demonstrate that the use of the CM system, together with the development security measures, allow only authorised changes to be made to the TOE.**

ACM_CAP.5.19C **The CM documentation shall demonstrate that the use of the integration procedures ensures that the generation of the TOE is correctly performed in an authorised manner.**

ACM_CAP.5.20C **The CM documentation shall demonstrate that the CM system is sufficient to ensure that the person responsible for accepting a configuration item into CM is not the person who developed it.**

ACM_CAP.5.21C **The CM documentation shall justify that the acceptance procedures provide for an adequate and appropriate review of changes to all configuration items.**

Evaluator action elements:

ACM_CAP.5.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

8.3 CM scope (ACM_SCP)

Objectives

The objective of this family is to ensure that all necessary TOE configuration items are tracked by the CM system. This helps to ensure that the integrity of these configuration items is protected through the capabilities of the CM system.

The objectives of this family include the following:

a) ensuring that the TOE implementation representation is tracked;

b) ensuring that all necessary documentation, including problem reports, are tracked during development and operation;

c) ensuring that configuration options (e.g. compiler switches) are tracked; and

d) ensuring that development tools are tracked.

Component levelling

The components in this family are levelled on the basis of which of the following are tracked by the CM system: the TOE implementation representation; design documentation; test documentation; user documentation; administrator documentation; CM documentation; security flaws; and development tools.

Application notes

ACM_SCP.1.1C introduces the requirement that the TOE implementation representation be tracked by the CM system. The TOE implementation representation refers to all hardware, software, and firmware that comprise the physical TOE. In the case of a software-only TOE, the implementation representation may consist solely of source and object code.

ACM_SCP.1.1C also introduces the requirement that the CM documentation be tracked by the CM system. This includes the CM plan, as well as information on the current versions of any tools that comprise the CM system.

ACM_SCP.2.1C introduces the requirement that security flaws be tracked by the CM system. This requires that information regarding previous security flaws and their resolution be maintained, as well as details regarding current security flaws.

ACM_SCP.3.1C introduces the requirement that development tools and other related information be tracked by the CM system. Examples of development tools are programming languages and compilers. Information pertaining to TOE generation items (such as compiler options, installation/ generation options, and build options) is an example of information relating to development tools.

ACM_SCP.1 TOE CM coverage

Objectives

A CM system can control changes only to those items that have been placed under CM. Placing the TOE implementation representation, design, tests, user and administrator documentation, and CM documentation under CM provides assurance that they have been modified in a controlled manner with proper authorisations.

Dependencies:

ACM_CAP.3 Authorisation controls

Developer action elements:

ACM_SCP.1.1D **The developer shall provide CM documentation.**

Content and presentation of evidence elements:

ACM_SCP.1.1C **The CM documentation shall show that the CM system, as a minimum, tracks the following: the TOE implementation representation, design documentation, test documentation, user documentation, administrator documentation, and CM documentation.**

ACM_SCP.1.2C **The CM documentation shall describe how configuration items are tracked by the CM system.**

Evaluator action elements:

ACM_SCP.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ACM_SCP.2 Problem tracking CM coverage

Objectives

A CM system can control changes only to those items that have been placed under CM. Placing the TOE implementation representation, design, tests, user and administrator documentation, and CM documentation under CM provides assurance that they have been modified in a controlled manner with proper authorisations.

The ability to track security flaws under CM ensures that security flaw reports are not lost or forgotten, and allows a developer to track security flaws to their resolution.

Dependencies:

ACM_CAP.3 Authorisation controls

Developer action elements:

ACM_SCP.2.1D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_SCP.2.1C The CM documentation shall show that the CM system, as a minimum, tracks the following: the TOE implementation representation, design documentation, test documentation, user documentation, administrator documentation, CM documentation**, and security flaws.**

ACM_SCP.2.2C The CM documentation shall describe how configuration items are tracked by the CM system.

Evaluator action elements:

ACM_SCP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ACM_SCP.3 Development tools CM coverage

Objectives

A CM system can control changes only to those items that have been placed under CM. Placing the TOE implementation representation, design, tests, user and administrator documentation, and CM documentation under CM provides assurance that they have been modified in a controlled manner with proper authorisations.

The ability to track security flaws under CM ensures that security flaw reports are not lost or forgotten, and allows a developer to track security flaws to their resolution.

Development tools play an important role in ensuring the production of a quality version of the TSF. Therefore, it is important to control modifications to these tools.

Dependencies:

ACM_CAP.3 Authorisation controls

Developer action elements:

ACM_SCP.3.1D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_SCP.3.1C The CM documentation shall show that the CM system, as a minimum, tracks the following: the TOE implementation representation, design documentation, test documentation, user documentation, administrator documentation, CM documentation, security flaws**, and development tools and related information.**

ACM_SCP.3.2C The CM documentation shall describe how configuration items are tracked by the CM system.

Evaluator action elements:

ACM_SCP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

9 Class ADO: Delivery and operation

Delivery and operation provides requirements for correct delivery, installation, generation, and start-up of the TOE.

Figure 9.1 shows the families within this class, and the hierarchy of components within the families.

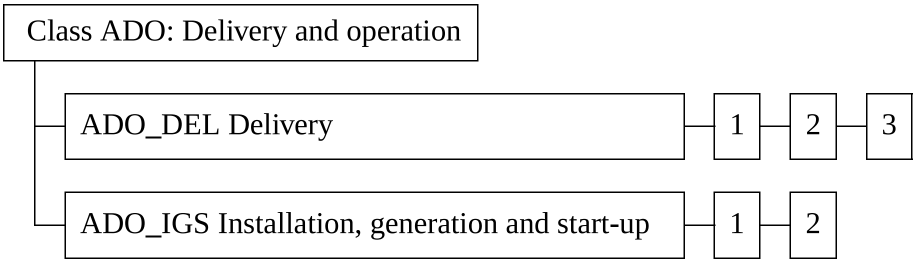
Class ADO: Delivery and operation
ADO_DEL Delivery
1
2
3
ADO_IGS Installation, generation and start-up
1
2

Figure 9.1 -Delivery and operation class decomposition

9.1 Delivery (ADO_DEL)

Objectives

The requirements for delivery call for system control and distribution facilities and procedures that provide assurance that the recipient receives the TOE that the sender intended to send, without any modifications. For a valid delivery, what is received must correspond precisely to the TOE master copy, thus avoiding any tampering with the actual version, or substitution of a false version.

Component levelling

The components in this family are levelled on the basis of increasing requirements on the developer to detect and prevent modifications to the TOE during delivery.

ADO_DEL.1 Delivery procedures

Dependencies:

No dependencies.

Developer action elements:

ADO_DEL.1.1D **The developer shall document procedures for delivery of the TOE or parts of it to the user.**

ADO_DEL.1.2D **The developer shall use the delivery procedures.**

Content and presentation of evidence elements:

ADO_DEL.1.1C **The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.**

Evaluator action elements:

ADO_DEL.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADO_DEL.2 Detection of modification

Dependencies:

ACM_CAP.3 Authorisation controls

Developer action elements:

ADO_DEL.2.1D The developer shall document procedures for delivery of the TOE or parts of it to the user.

ADO_DEL.2.2D The developer shall use the delivery procedures.

Content and presentation of evidence elements:

ADO_DEL.2.1C The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.

ADO_DEL.2.2C **The delivery documentation shall describe how the various procedures and technical measures provide for the detection of modifications, or any discrepancy between the developer's master copy and the version received at the user site.**

ADO_DEL.2.3C **The delivery documentation shall describe how the various procedures allow detection of attempts to masquerade as the developer, even in cases in which the developer has sent nothing to the user's site.**

Evaluator action elements:

ADO_DEL.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO_DEL.3 Prevention of modification

Dependencies:

ACM_CAP.3 Authorisation controls

Developer action elements:

ADO_DEL.3.1D The developer shall provide document procedures for delivery of the TOE or parts of it to the user.

ADO_DEL.3.2D The developer shall use the delivery procedures.

Content and presentation of evidence elements:

ADO_DEL.3.1C The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.

ADO_DEL.3.2C The delivery documentation shall describe how the various procedures and technical measures provide for the **prevention** of modifications, or any discrepancy between the developer's master copy and the version received at the user site.

ADO_DEL.3.3C The delivery documentation shall describe how the various procedures allow detection of attempts to masquerade as the developer, even in cases in which the developer has sent nothing to the user's site.

Evaluator action elements:

ADO_DEL.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

9.2 Installation, generation and start-up (ADO_IGS)

Objectives

Installation, generation, and start-up procedures are useful for ensuring that the TOE has been installed, generated, and started up in a secure manner as intended by the developer. The requirements for installation, generation and start-up call for a secure transition from the TOE's implementation representation being under configuration control to its initial operation in the user environment.

Component levelling

The components in this family are levelled on the basis of whether the TOE generation options are logged.

Application notes

It is recognised that the application of these requirements will vary depending on aspects such as whether the TOE is an IT-product or -system, whether it is delivered in an operational state, or whether it has to be brought up at the TOE owner's site, etc. For a given TOE, there will normally be a division of responsibility with respect to installation, generation and start-up between the TOE developer and the owner of the TOE, but there are examples that where all activities take place at one site. For example, for a smart card all aspects of installation, generation and start-up may have been performed at the TOE developer's site. On the other hand the TOE might be delivered as an IT-system in the form of software, where all aspects of installation, generation and start-up are carried out at the TOE owner's site.

It might also be the case that the TOE is already installed by the time the evaluation starts. In this case it may be inappropriate to demand and analyse installation procedures.

Furthermore, the generation requirements are applicable only to TOEs that provide the ability to generate portions of an operational TOE from its implementation representation.

The installation, generation, and start-up procedures may exist as a separate documents or could be grouped with other administrative guidance. The requirements in this assurance family are presented separately from those in the AGD_ADM family, due to the infrequent, possibly one-time use of the installation, generation and start-up procedures.

ADO_IGS.1 Installation, generation, and start-up procedures

Dependencies:

AGD_ADM.1 Administrator guidance

Developer action elements:

ADO_IGS.1.1D **The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.**

Content and presentation of evidence elements:

ADO_IGS.1.1C **The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.**

Evaluator action elements:

ADO_IGS.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADO_IGS.1.2E **The evaluator shall determine that the installation, generation, and start-up procedures result in a secure configuration.**

ADO_IGS.2 Generation log

Dependencies:

AGD_ADM.1 Administrator guidance

Developer action elements:

ADO_IGS.2.1D The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.

Content and presentation of evidence elements:

ADO_IGS.2.1C The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.

ADO_IGS.2.2C **The documentation shall describe procedures capable of creating a log containing the generation options used to generate the TOE in such a way that it is possible to determine exactly how and when the TOE was generated.**

Evaluator action elements:

ADO_IGS.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence**.**

ADO_IGS.2.2E The evaluator shall determine that the installation, generation, and start-up procedures result in a secure configuration.

10 Class ADV: Development

The development class encompasses four families of requirements for representing the TSF at various levels of abstraction from the functional interface to the implementation representation. The development class also includes a family of requirements for a correspondence mapping between the various TSF representations, ultimately requiring a demonstration of correspondence from the least abstract representation through all intervening representations to the TOE summary specification provided in the ST. In addition, there is a family of requirements for a TSP model, and for correspondence mappings between the TSP, the TSP model, and the functional specification. Finally, there is a family of requirements on the internal structure of the TSF, which covers aspects such as modularity, layering, and minimisation of complexity.

Figure 10.1 shows the families within this class, and the hierarchy of components within the families.

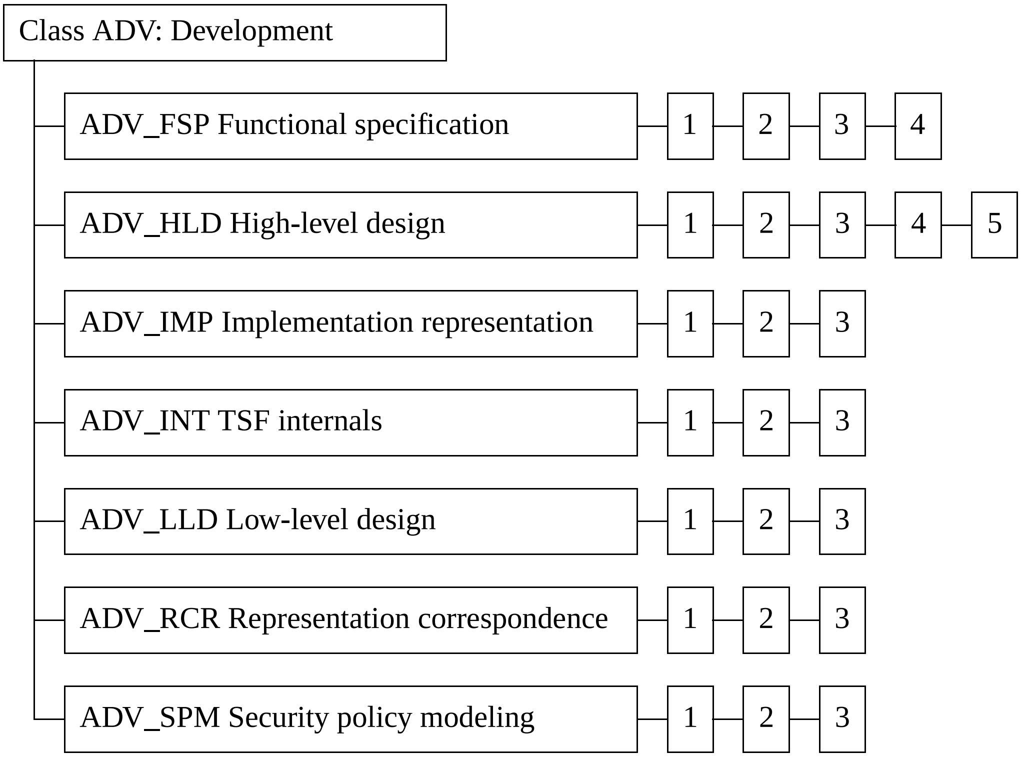
Class ADV: Development
ADV_FSP Functional specification
1
2
3
4
ADV_HLD High-level design
1
2
3
4
5
ADV_IMP Implementation representation
1
2
3
ADV_INT TSF internals
1
2
3
ADV_LLD Low-level design
1
2
3
ADV_RCR Representation correspondence
1
2
3
ADV_SPM Security policy modeling
1
2
3

Figure 10.1 - Development class decomposition

The paradigm evident for these families is one of a functional specification of the TSF, decomposing the TSF into subsystems, decomposing the subsystems into modules, showing the implementation of the modules, and demonstration of correspondence between all decompositions that are provided as evidence. The requirements for the various TSF representations are separated into different families, however, to allow the PP/ST author to specify which subset of the TSF representations are required.

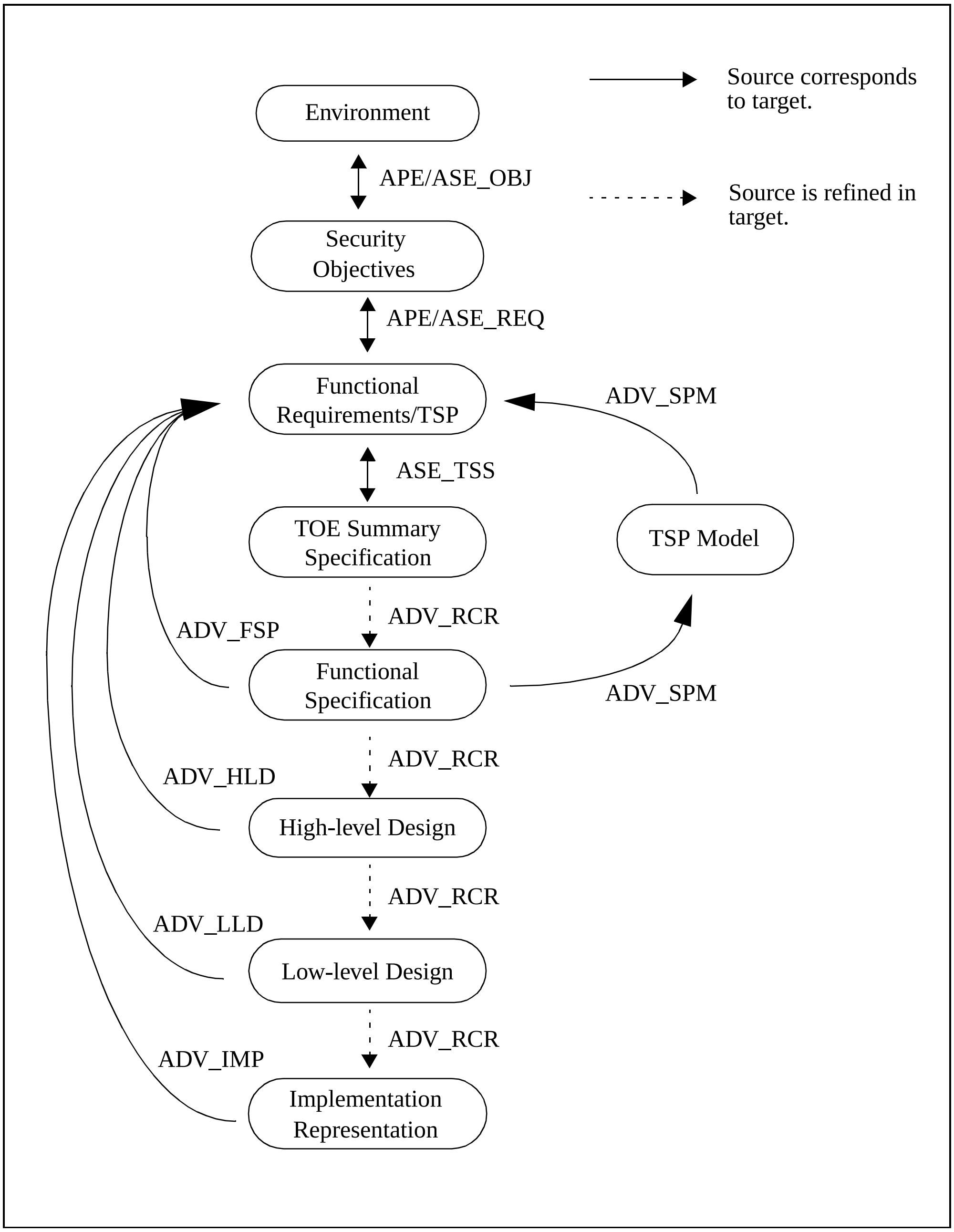

Figure 10.2 - Relationships between TOE representations and requirements

Figure 10.2 indicates the relationships between the various TSF representations and the objectives and requirements that they are intended to address. As the figure indicates, the APE and ASE classes define the requirements for the correspondence between the functional requirements and the IT security objectives as well as between the IT security objectives and the TOE's anticipated environment. Class ASE also defines requirements for the correspondence between both the IT security objectives and functional requirements and the TOE summary specification.

The requirements for all other correspondence shown in Figure 10.2 are defined in the ADV class. The ADV_SPM family defines the requirements for correspondence between the TSP and the TSP model, and between the TSP model and the functional specification. The ADV_RCR family defines the requirements for correspondence between all available TSF representations from the TOE summary specification through the implementation representation. Finally, each assurance family specific to a TSF representation (i.e. ADV_FSP, ADV_HLD, ADV_LLD and ADV_IMP) defines requirements relating that TSF representation to the functional requirements, the combination of which helps to ensure that the TOE security functional requirements have been addressed. The traceability analysis is always to be performed from the highest-level TSF representation down through each of the TSF representations that are provided. The CC captures this traceability requirement via dependencies on the ADV_RCR family. The ADV_INT family is not represented in this figure, as it is related to the internal structure of the TSF, and is only indirectly related to the process of refinement of the TSF representations.

Application notes

The TOE security policy (TSP) is the set of rules that regulate how resources are managed, protected and distributed within a TOE, expressed by the TOE security functional requirements. The developer is not explicitly required to provide a TSP, as the TSP is expressed by the TOE security functional requirements, through a combination of security function policies (SFPs) and the other individual requirement elements.

The TOE security functions (TSF) are all the parts of the TOE that have to be relied upon for enforcement of the TSP. The TSF includes both functions that directly enforce the TSP, and also those functions that, while not directly enforcing the TSP, contribute to the enforcement of the TSP in a more indirect manner.

Although the requirements within the ASE_TSS family and within several families of this class call for several different TSF representations, it is not absolutely necessary for each and every TSF representation to be in a separate document. Indeed, it may be the case that a single document meets the documentation requirements for more than one TSF representation, since it is the information about each of these TSF representations that is required, rather than the resulting document structure. In cases where multiple TSF representations are combined within a single document, the developer should indicate which documents meet which requirements.

Three types of specification style are mandated by this class: informal, semiformal and formal. The functional specification, high-level design, low-level design and TSP models will be written using one or more of these specification styles. Ambiguity in these specifications is reduced by using an increased level of formality.

An informal specification is written as prose in natural language. Natural language is used here as meaning communication in any commonly spoken tongue (e.g. Dutch, English, French, German). An informal specification is not subject to any notational or special restrictions other than those required as ordinary conventions for that language (e.g. grammar and syntax). While no notational restrictions apply, the informal specification is also required to provide defined meanings for terms that are used in a context other than that accepted by normal usage.

A semiformal specification is written in a restricted syntax language and is typically accompanied by supporting explanatory (informal) prose. The restricted syntax language may be a natural language with restricted sentence structure and keywords with special meanings, or it may be diagrammatic (e.g. data-flow diagrams, state transition diagrams, entity-relationship diagrams, data structure diagrams, and process or program structure diagrams). Whether based on diagrams or natural language, a set of conventions must be supplied to define the restrictions placed on the syntax.

A formal specification is written in a notation based upon well-established mathematical concepts, and is typically accompanied by supporting explanatory (informal) prose. These mathematical concepts are used to define the syntax and semantics of the notation and the proof rules that support logical reasoning. The syntactic and semantic rules supporting a formal notation should define how to recognise constructs unambiguously and determine their meaning. There needs to be evidence that it is impossible to derive contradictions, and all rules supporting the notation need to be defined or referenced.

Significant assurance can be gained by ensuring that the TSF can be traced though each of its representations, and by ensuring that the TSP model corresponds to the functional specification. The ADV_RCR family contains requirements for correspondence mappings between the various TSF representations, and the ADV_SPM family contains requirements for a correspondence mapping between the TSP model and the functional specification. A correspondence can take the form of an informal demonstration, a semiformal demonstration, or a formal proof.

When an informal demonstration of correspondence is required, this means that only a basic correspondence is required. Correspondence methods include, for example, the use of a two-dimensional table with entries denoting correspondence, or the use of appropriate notation of design diagrams. Pointers and references to other documents may also be used.

A semiformal demonstration of correspondence requires a structured approach at the analysis of the correspondence. This approach should lessen ambiguity that could exist in an informal correspondence by limiting the interpretation of the terms included in the correspondence. Pointers and references to other documents may be used.

A formal proof of correspondence requires that well-established mathematical concepts be used to define the syntax and semantics of the formal notation and the proof rules that support logical reasoning. The security properties need to be expressible in the formal specification language, and these security properties need to be shown to be satisfied by the formal specification. Pointers and references to other documents may also be used.

The ADV_RCR.*.1C elements require that the developer provide evidence, for each adjacent pair of TSF representations, that all relevant security functionality of the more abstract TSF representation is refined in the less abstract TSF representation. The ADV_FSP.*.2E, ADV_HLD.*.2E, ADV_LLD.*.2E and ADV_IMP.*.2E elements each require the evaluator to determine that the TSF represented by that family of requirements is an accurate and complete instantiation of the TOE security functional requirements. In order to determine that a TSF representation is an accurate and complete instantiation of the TOE security functional requirements, it is intended that the evaluator use the evidence provided by the developer in ADV_RCR.*.1C as an input to this determination. By establishing a correspondence between the TOE security functional requirements and each of successive TSF representations down the chain, this step-wise process will ultimately provide more assurance that the least abstract TSF representation corresponds to the TOE security functional requirements, which is the ultimate goal of this class. If the evaluator makes no correspondence determinations back to the TOE security functional requirements for intermediate TSF representations, then trying to determine the correspondence from the least abstract TSF representation back to the TOE security functional requirements may represent too large a step to be accurately performed. Finally, depending on the set of TSF representations that are required, it is quite possible that the low-level design, high-level design, or even the functional specification might be the least abstract TSF representation that is provided.

10.1 Functional specification (ADV_FSP)

Objectives

The functional specification is a high-level description of the user-visible interface and behaviour of the TSF. It is an instantiation of the TOE security functional requirements. The functional specification has to show that all the TOE security functional requirements are addressed.

Component levelling

The components in this family are levelled on the basis of the degree of formalism required of the functional specification, and the degree of detail provided for the external interfaces to the TSF.

Application notes

The ADV_FSP.*.2E elements within this family define a requirement that the evaluator determine that the functional specification is an accurate and complete instantiation of the TOE security functional requirements. This provides a direct correspondence between the TOE security functional requirements and the functional specification, in addition to the pairwise correspondences required by the ADV_RCR family. It is expected that the evaluator will use the evidence provided in ADV_RCR as an input to making this determination, and the requirement for completeness is intended to be relative to the level of abstraction of the functional specification.

For ADV_FSP.1.3C, it is intended that sufficient information is provided in the functional specification to understand how the TOE security functional requirements have been addressed, and to enable the specification of tests which reflect the TOE security functional requirements in the ST. It is not necessarily the case that such testing will cover all possible return values and error messages which could be generated at the interface, but the information provided should make clear the results of using an interface in the case of success and the most common instances of failure.

ADV_FSP.2.3C introduces a requirement for a complete presentation of the functional interface is required. This will provide the necessary detail for supporting both thorough testing of the TOE and the assessment of vulnerabilities.

In the context of the level of formality of the functional specification, informal, semiformal and formal are considered to be hierarchical in nature. Thus, ADV_FSP.1.1C and ADV_FSP.2.1C may also be met with either a semiformal or formal functional specification, provided that it is supported by informal, explanatory text where appropriate. In addition, ADV_FSP.3.1C may also be met with a formal functional specification.

ADV_FSP.1 Informal functional specification

Dependencies:

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_FSP.1.1D **The developer shall provide a functional specification.**

Content and presentation of evidence elements:

ADV_FSP.1.1C **The functional specification shall describe the TSF and its external interfaces using an informal style.**

ADV_FSP.1.2C The functional specification shall be internally consistent.

ADV_FSP.1.3C **The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing details of effects, exceptions and error messages, as appropriate.**

ADV_FSP.1.4C The functional specification shall completely represent the TSF.

Evaluator action elements:

ADV_FSP.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADV_FSP.1.2E **The evaluator shall determine that the functional specification is an accurate and complete instantiation of the TOE security functional requirements.**

ADV_FSP.2 Fully defined external interfaces

Dependencies:

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_FSP.2.1D The developer shall provide a functional specification.

Content and presentation of evidence elements:

ADV_FSP.2.1C The functional specification shall describe the TSF and its external interfaces using an informal style.

ADV_FSP.2.2C The functional specification shall be internally consistent.

ADV_FSP.2.3C The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing **complete** details of **all** effects, exceptions and error messages.

ADV_FSP.2.4C The functional specification shall completely represent the TSF.

ADV_FSP.2.5C The functional specification shall include rationale that the TSF is completely represented.

Evaluator action elements:

ADV_FSP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.2.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the TOE security functional requirements.

ADV_FSP.3 Semiformal functional specification

Dependencies:

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_FSP.3.1D The developer shall provide a functional specification.

Content and presentation of evidence elements:

ADV_FSP.3.1C The functional specification shall describe the TSF and its external interfaces using a **semiformal** style**, supported by informal, explanatory text where appropriate.**

ADV_FSP.3.2C The functional specification shall be internally consistent.

ADV_FSP.3.3C The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing complete details of all effects, exceptions and error messages.

ADV_FSP.3.4C The functional specification shall completely represent the TSF.

ADV_FSP.3.5C The functional specification shall include rationale that the TSF is completely represented.

Evaluator action elements:

ADV_FSP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.3.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the TOE security functional requirements.

ADV_FSP.4 Formal functional specification

Dependencies:

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_FSP.4.1D The developer shall provide a functional specification.

Content and presentation of evidence elements:

ADV_FSP.4.1C The functional specification shall describe the TSF and its external interfaces using a **formal** style, supported by informal, explanatory text where appropriate.

ADV_FSP.4.2C The functional specification shall be internally consistent.

ADV_FSP.4.3C The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing complete details of all effects, exceptions and error messages.

ADV_FSP.4.4C The functional specification shall completely represent the TSF.

ADV_FSP.4.5C The functional specification shall include rationale that the TSF is completely represented.

Evaluator action elements:

ADV_FSP.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.4.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the TOE security functional requirements.

10.2 High-level design (ADV_HLD)

Objectives

The high-level design of a TOE provides a description of the TSF in terms of major structural units (i.e. subsystems) and relates these units to the functions that they provide. The high-level design requirements are intended to provide assurance that the TOE provides an architecture appropriate to implement the claimed functional requirements.

The high-level design refines the functional specification into subsystems. For each subsystem of the TSF, the high-level design describes its purpose and function, and identifies the security functions contained in the subsystem. The interrelationships of all subsystems are also defined in the high-level design. These interrelationships will be represented as external interfaces for data flow, control flow, etc., as appropriate.

Component levelling

The components in this family are levelled on the basis of the degree of formalism required of the high-level design, and on the degree of detail required for the interface specifications.

Application notes

The developer is expected to describe the design of the TSF in terms of subsystems. The term “subsystem” is used here to express the idea of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have “subsystems”, the developer is expected to represent a similar level of decomposition. For example, a design may be similarly decomposed using “layers”, “domains”, or “servers”.

The term “security functionality” is used to represent the set of operations that a subsystem performs in contribution to security functions implemented by the TOE. This distinction is made because design constructs, such as subsystems and modules, do not necessarily relate to specific security functions. While a given subsystem may correspond directly to a security function, or even multiple security functions, it is also possible that many subsystems must be combined to implement a single security function.

The term “TSP-enforcing subsystem” refers to a subsystem that contributes to the enforcement of the TSP, either directly or indirectly.

The ADV_HLD.*.2E elements within this family define a requirement that the evaluator determine that the high-level design is an accurate and complete instantiation of the TOE security functional requirements. This provides a direct correspondence between the TOE security functional requirements and the high-level design, in addition to the pairwise correspondences required by the ADV_RCR family. It is expected that the evaluator will use the evidence provided in ADV_RCR as an input to making this determination, and the requirement for completeness is intended to be relative to the level of abstraction of the high-level design.

ADV_HLD.3.8C introduces a requirement for a complete presentation for the interfaces to the subsystems. This will provide the necessary detail for supporting both thorough testing of the TOE (using components from ATE_DPT), and the assessment of vulnerabilities.

In the context of the level of formality of the high-level design, informal, semiformal and formal are considered to be hierarchical in nature. Thus, ADV_HLD.1.1C and ADV_HLD.2.1C may also be met with either a semiformal or formal high-level design, and ADV_HLD.3.1C and ADV_HLD.4.1C may also be met with a formal high-level design.

ADV_HLD.1 Descriptive high-level design

Dependencies:

ADV_FSP.1 Informal functional specification

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_HLD.1.1D **The developer shall provide the high-level design of the TSF.**

Content and presentation of evidence elements:

ADV_HLD.1.1C **The presentation of the high-level design shall be informal.**

ADV_HLD.1.2C The high-level design shall be internally consistent.

ADV_HLD.1.3C **The high-level design shall describe the structure of the TSF in terms of subsystems.**

ADV_HLD.1.4C **The high-level design shall describe the security functionality provided by each subsystem of the TSF.**

ADV_HLD.1.5C **The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.**

ADV_HLD.1.6C **The high-level design shall identify all interfaces to the subsystems of the TSF.**

ADV_HLD.1.7C The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

Evaluator action elements:

ADV_HLD.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADV_HLD.1.2E **The evaluator shall determine that the high-level design is an accurate and complete instantiation of the TOE security functional requirements.**

ADV_HLD.2 Security enforcing high-level design

Dependencies:

ADV_FSP.1 Informal functional specification

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_HLD.2.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements:

ADV_HLD.2.1C The presentation of the high-level design shall be informal.

ADV_HLD.2.2C The high-level design shall be internally consistent.

ADV_HLD.2.3C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.2.4C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.2.5C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.2.6C The high-level design shall identify all interfaces to the subsystems of the TSF.

ADV_HLD.2.7C The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

ADV_HLD.2.8C The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing details of effects, exceptions and error messages, as appropriate.

ADV_HLD.2.9C **The high-level design shall describe the separation of the TSF into TSP-enforcing and other subsystems.**

Evaluator action elements:

ADV_HLD.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.2.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of the TOE security functional requirements.

ADV_HLD.3 Semiformal high-level design

Dependencies:

ADV_FSP.3 Semiformal functional specification

ADV_RCR.2 Semiformal correspondence demonstration

Developer action elements:

ADV_HLD.3.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements:

ADV_HLD.3.1C The presentation of the high-level design shall be **semiformal**.

ADV_HLD.3.2C The high-level design shall be internally consistent.

ADV_HLD.3.3C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.3.4C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.3.5C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.3.6C The high-level design shall identify all interfaces to the subsystems of the TSF.

ADV_HLD.3.7C The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

ADV_HLD.3.8C The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing **complete** details of **all** effects, exceptions and error messages.

ADV_HLD.3.9C The high-level design shall describe the separation of the TSF into TSP-enforcing and other subsystems.

Evaluator action elements:

ADV_HLD.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.3.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of the TOE security functional requirements.

ADV_HLD.4 Semiformal high-level explanation

Dependencies:

ADV_FSP.3 Semiformal functional specification

ADV_RCR.2 Semiformal correspondence demonstration

Developer action elements:

ADV_HLD.4.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements:

ADV_HLD.4.1C The presentation of the high-level design shall be semiformal.

ADV_HLD.4.2C The high-level design shall be internally consistent.

ADV_HLD.4.3C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.4.4C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.4.5C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.4.6C The high-level design shall identify all interfaces to the subsystems of the TSF.

ADV_HLD.4.7C The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

ADV_HLD.4.8C The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing complete details of all effects, exceptions and error messages.

ADV_HLD.4.9C The high-level design shall describe the separation of the TSF into TSP-enforcing and other subsystems.

ADV_HLD.4.10C **The high-level design shall justify that the identified means of achieving separation, including any protection mechanisms, are sufficient to ensure a clear and effective separation of TSP-enforcing from non-TSP-enforcing functions.**

ADV_HLD.4.11C **The high-level design shall justify that the TSF mechanisms are sufficient to implement the security functions identified in the high-level design.**

Evaluator action elements:

ADV_HLD.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.4.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of the TOE security functional requirements.

ADV_HLD.5 Formal high-level design

Dependencies:

ADV_FSP.4 Formal functional specification

ADV_RCR.3 Formal correspondence demonstration

Developer action elements:

ADV_HLD.5.1D The developer shall provide the high-level design of the TSF.

Content and presentation of evidence elements:

ADV_HLD.5.1C The presentation of the high-level design shall be **formal**.

ADV_HLD.5.2C The high-level design shall be internally consistent.

ADV_HLD.5.3C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.5.4C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.5.5C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.5.6C The high-level design shall identify all interfaces to the subsystems of the TSF.

ADV_HLD.5.7C The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

ADV_HLD.5.8C The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing complete details of all effects, exceptions and error messages.

ADV_HLD.5.9C The high-level design shall describe the separation of the TSF into TSP-enforcing and other subsystems.

ADV_HLD.5.10C The high-level design shall justify that the identified means of achieving separation, including any protection mechanisms, are sufficient to ensure a clear and effective separation of TSP-enforcing from non-TSP-enforcing functions.

ADV_HLD.5.11C The high-level design shall justify that the TSF mechanisms are sufficient to implement the security functions identified in the high-level design.

Evaluator action elements:

ADV_HLD.5.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.5.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of the TOE security functional requirements.

10.3 Implementation representation (ADV_IMP)

Objectives

The description of the implementation representation in the form of source code, firmware, hardware drawings, etc. captures the detailed internal workings of the TSF in support of analysis.

Component levelling

The components in this family are levelled on the basis of the completeness and structure of the implementation representation provided.

Application notes

The implementation representation is used to express the notion of the least abstract representation of the TSF, specifically the one that is used to create the TSF itself without further design refinement. Source code that is then compiled or a hardware drawing that is used to build the actual hardware are examples of parts of an implementation representation.

It is possible that evaluators may use the implementation representation to directly support other evaluation activities (e.g. vulnerability analysis, test coverage analysis, or identification of additional evaluator tests). It is expected that PP/ST authors will select a component that requires that the implementation is complete and comprehensive enough to address the needs of all other requirements included in the PP/ST.

ADV_IMP.1 Subset of the implementation of the TSF

Application notes

ADV_IMP.1.1D requires that the developer provide the implementation representation for a subset of the TSF. The intention is that access to at least a portion of the TSF will provide the evaluator with an opportunity to examine the implementation representation for those portions of the TOE where such an examination can add significantly to the understanding of, and assurance in, the mechanisms employed. Provision of a sample of the implementation representation will also allow the evaluator to sample the traceability evidence to gain assurance in the approach taken for refinement, and to assess the presentation of the implementation representation itself.

ADV_IMP.1.2E element defines a requirement that the evaluator determine that the least abstract TSF representation is an accurate and complete instantiation of the TOE security functional requirements. This provides a direct correspondence between the TOE security functional requirements and the least abstract TSF representation, in addition to the pairwise correspondences required by the ADV_RCR family. It is expected that the evaluator will use the evidence provided in ADV_RCR as an input to making this determination. The least abstract TSF representation for this component is an aggregate of the implementation representation that is provided and that portion of the low-level design for which no corresponding implementation representation is provided.

Dependencies:

ADV_LLD.1 Descriptive low-level design

ADV_RCR.1 Informal correspondence demonstration

ALC_TAT.1 Well-defined development tools

Developer action elements:

ADV_IMP.1.1D **The developer shall provide the implementation representation for a selected subset of the TSF.**

Content and presentation of evidence elements:

ADV_IMP.1.1C **The implementation representation shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.**

ADV_IMP.1.2C The implementation representation shall be internally consistent.

Evaluator action elements:

ADV_IMP.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADV_IMP.1.2E **The evaluator shall determine that the least abstract TSF representation provided is an accurate and complete instantiation of the TOE security functional requirements.**

ADV_IMP.2 Implementation of the TSF

Application notes

The ADV_IMP.2.2E element defines a requirement that the evaluator determine that the implementation representation is an accurate and complete instantiation of the TOE security functional requirements. This provides a direct correspondence between the TOE security functional requirements and the implementation representation, in addition to the pairwise correspondences required by the ADV_RCR family. It is expected that the evaluator will use the evidence provided in ADV_RCR as an input to making this determination.

Dependencies:

ADV_LLD.1 Descriptive low-level design

ADV_RCR.1 Informal correspondence demonstration

ALC_TAT.1 Well-defined development tools

Developer action elements:

ADV_IMP.2.1D The developer shall provide the implementation representation for **the entire TSF**.

Content and presentation of evidence elements:

ADV_IMP.2.1C The implementation representation shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.

ADV_IMP.2.2C The implementation representation shall be internally consistent.

ADV_IMP.2.3C **The implementation representation shall describe the relationships between all portions of the implementation.**

Evaluator action elements:

ADV_IMP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_IMP.2.2E The evaluator shall determine that the **implementation representation** is an accurate and complete instantiation of the TOE security functional requirements.

ADV_IMP.3 Structured implementation of the TSF

Application notes

The ADV_IMP.3.2E element defines a requirement that the evaluator determine that the implementation representation is an accurate and complete instantiation of the TOE security functional requirements. This provides a direct correspondence between the TOE security functional requirements and the implementation representation, in addition to the pairwise correspondences required by the ADV_RCR family. It is expected that the evaluator will use the evidence provided in ADV_RCR as an input to making this determination.

Dependencies:

ADV_INT.1 Modularity

ADV_LLD.1 Descriptive low-level design

ADV_RCR.1 Informal correspondence demonstration

ALC_TAT.1 Well-defined development tools

Developer action elements:

ADV_IMP.3.1D The developer shall provide the implementation representation for the entire TSF.

Content and presentation of evidence elements:

ADV_IMP.3.1C The implementation representation shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.

ADV_IMP.3.2C The implementation representation shall be internally consistent.

ADV_IMP.3.3C The implementation representation shall describe the relationships between all portions of the implementation.

ADV_IMP.3.4C The implementation representation shall be structured into small and comprehensible sections.

Evaluator action elements:

ADV_IMP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_IMP.3.2E The evaluator shall determine that the implementation representation is an accurate and complete instantiation of the TOE security functional requirements.

10.4 TSF internals (ADV_INT)

Objectives

This family addresses the internal structure of the TSF. Requirements are presented for modularity, layering (to separate levels of abstraction and minimise circular dependencies), minimisation of the complexity of policy enforcement mechanisms, and the minimisation of the amount of non-TSP-enforcing functionality within the TSF — thus resulting in a TSF that is simple enough to be analysed.

Modular design reduces the interdependence between elements of the TSF and thus reduces the risk that a change or error in one module will have effects throughout the TOE. Thus, a modular design provides the basis for determining the scope of interaction with other elements of the TSF, provides for increased assurance that unexpected effects do not occur, and also provides the basis for designing and evaluating test suites.

The use of layering and of simpler designs for the TSP-enforcing functionality reduces the complexity of the TSF. This in turn enables a better understanding of the TSF, providing more assurance that the TOE security functional requirements are accurately and completely instantiated in the implementation.

Minimising the amount of functionality in the TSF that does not enforce the TSP, reduces the possibility of flaws in the TSF. In combination with modularity and layering, it allows the evaluator to focus only on that functionality which is necessary for TSP enforcement.

Design complexity minimisation contributes to the assurance that the code is understood — the less complex the code in the TSF, the greater the likelihood that the design of the TSF is comprehensible. Design complexity minimisation is a key characteristic of a reference validation mechanism.

Component levelling

The components in this family are levelled on the basis of the amount of structure and minimisation required.

Application notes

The term “portions of the TSF” is used to represent parts of the TSF with a varying granularity based on the available TSF representations. The functional specification allows identification in terms of interfaces, the high-level design allows identification in terms of subsystems, the low-level design allows identification in terms of modules, and the implementation representation allows identification in terms of implementation units (e.g. source code files).

The ADV_INT.2.5C and ADV_INT.3.5C elements address minimisation of mutual interactions between layers. Nevertheless, it is still permissible to have mutual interactions between layers, but in such cases the developer is required to demonstrate that these mutual interactions are necessary and cannot reasonably be avoided.

Several of the elements within the components for this family refer to the architectural description. The architectural description is at a similar level of abstraction to the low-level design, in that it is concerned with the modules of the TSF. Whereas the low-level design describes the design of the modules of the TSF, the purpose of the architectural description is to provide evidence of modularity, layering, and minimisation of complexity of the TSF, as applicable. Both the low-level design and the implementation representation are required to be in compliance with the architectural description, to provide assurance that these TSF representations possess the required modularity, layering, and minimisation of complexity.

ADV_INT.1 Modularity

Dependencies:

ADV_IMP.1 Subset of the implementation of the TSF

ADV_LLD.1 Descriptive low-level design

Developer action elements:

ADV_INT.1.1D **The developer shall design and structure the TSF in a modular fashion that avoids unnecessary interactions between the modules of the design.**

ADV_INT.1.2D **The developer shall provide an architectural description.**

Content and presentation of evidence elements:

ADV_INT.1.1C **The architectural description shall identify the modules of the TSF.**

ADV_INT.1.2C **The architectural description shall describe the purpose, interface, parameters, and effects of each module in the TSF.**

ADV_INT.1.3C **The architectural description shall describe how the TSF design provides for largely independent modules that avoid unnecessary interactions.**

Evaluator action elements:

ADV_INT.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADV_INT.1.2E **The evaluator shall determine that both the low-level design and the implementation representation are in compliance with the architectural description.**

ADV_INT.2 Reduction of complexity

Application notes

This component introduces a reference monitor concept by requiring the minimisation of complexity of the portions of the TSF that enforce the access control and/or information flow control policies identified in the TSP.

Dependencies:

ADV_IMP.1 Subset of the implementation of the TSF

ADV_LLD.1 Descriptive low-level design

Developer action elements:

ADV_INT.2.1D The developer shall design and structure the TSF in a modular fashion that avoids unnecessary interactions between the modules of the design.

ADV_INT.2.2D The developer shall provide an architectural description.

ADV_INT.2.3D The developer shall design and structure the TSF in a layered fashion that minimises mutual interactions between the layers of the design.

ADV_INT.2.4D The developer shall design and structure the TSF in such a way that minimises the complexity of the portions of the TSF that enforce any access control and/or information flow control policies.

Content and presentation of evidence elements:

ADV_INT.2.1C The architectural description shall identify the modules of the TSF **and shall specify which portions of the TSF enforce the access control and/or information flow control policies**.

ADV_INT.2.2C The architectural description shall describe the purpose, interface, parameters, and effects of each module of the TSF.

ADV_INT.2.3C The architectural description shall describe how the TSF design provides for largely independent modules that avoid unnecessary interactions.

ADV_INT.2.4C **The architectural description shall describe the layering architecture.**

ADV_INT.2.5C **The architectural description shall show that mutual interactions have been minimised, and justify those that remain.**

ADV_INT.2.6C **The architectural description shall describe how the portions of the TSF that enforce any access control and/or information flow control policies have been structured to minimise complexity.**

Evaluator action elements:

ADV_INT.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_INT.2.2E The evaluator shall determine that both the low-level design and the implementation representation are in compliance with the architectural description.

ADV_INT.3 Minimisation of complexity

Application notes

This component requires that the reference monitor property "simple enough to be analysed" is fully addressed. When this component is combined with the functional requirements FPT_RVM.1 and FPT_SEP.3, the reference monitor concept would be fully realised.

Dependencies:

ADV_IMP.2 Implementation of the TSF

ADV_LLD.1 Descriptive low-level design

Developer action elements:

ADV_INT.3.1D The developer shall design and structure the TSF in a modular fashion that avoids unnecessary interactions between the modules of the design.

ADV_INT.3.2D The developer shall provide an architectural description.

ADV_INT.3.3D The developer shall design and structure the TSF in a layered fashion that minimises mutual interactions between the layers of the design.

ADV_INT.3.4D The developer shall design and structure the TSF in such a way that minimises the complexity of the **entire TSF**.

ADV_INT.3.5D **The developer shall design and structure the portions of the TSF that enforce any access control and/or information flow control policies such that they are simple enough to be analysed.**

ADV_INT.3.6D The developer shall ensure that functions whose objectives are not relevant for the TSF are excluded from the TSF modules.

Content and presentation of evidence elements:

ADV_INT.3.1C The architectural description shall identify the modules of the TSF and shall specify which portions of the TSF enforce the access control and/or information flow control policies.

ADV_INT.3.2C The architectural description shall describe the purpose, interface, parameters, and side-effects of each module of the TSF.

ADV_INT.3.3C The architectural description shall describe how the TSF design provides for largely independent modules that avoid unnecessary interactions.

ADV_INT.3.4C The architectural description shall describe the layering architecture.

ADV_INT.3.5C The architectural description shall show that mutual interactions have been minimised, and justify those that remain.

ADV_INT.3.6C The architectural description shall describe how the **entire TSF has** been structured to minimise complexity.

ADV_INT.3.7C **The architectural description shall justify the inclusion of any non-TSP-enforcing modules in the TSF.**

Evaluator action elements:

ADV_INT.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_INT.3.2E The evaluator shall determine that both the low-level design and the implementation representation are in compliance with the architectural description.

ADV_INT.3.3E **The evaluator shall confirm that the portions of the TSF that enforce any access control and/or information flow control policies are simple enough to be analysed.**

10.5 Low-level design (ADV_LLD)

Objectives

The low-level design of a TOE provides a description of the internal workings of the TSF in terms of modules and their interrelationships and dependencies. The low-level design provides assurance that the TSF subsystems have been correctly and effectively refined.

For each module of the TSF, the low-level design describes its purpose, function, interfaces, dependencies, and the implementation of any TSP-enforcing functions.

Component levelling

The components in this family are levelled on the basis of the degree of formalism required of the low-level design, and on the degree of detail required for the interface specifications.

Application notes

The term “TSP-enforcing module” refers to any module that contributes to TSP enforcement.

The term “security functionality” is used to represent the set of operations that a module performs in contribution to security functions implemented by the TOE. This distinction is made because modules do not necessarily relate to specific security functions. While a given module may correspond directly to a security function, or even multiple security functions, it is also possible that many modules must be combined to implement a single security function.

The ADV_LLD.*.6C elements require that the low-level design describe how each TSP-enforcing function is provided. The intent of this requirement is that the low-level design provide a description of how each module is expected to be implemented from a design perspective.

The ADV_LLD.*.2E elements within this family define a requirement that the evaluator determine that the low-level design is an accurate and complete instantiation of the TOE security functional requirements. This provides a direct correspondence between the TOE security functional requirements and the low-level design, in addition to the pairwise correspondences required by the ADV_RCR family. It is expected that the evaluator will use the evidence provided in ADV_RCR as an input to making this determination, and the requirement for completeness is intended to be relative to the level of abstraction of the low-level design.

ADV_LLD.2.9C introduces a requirement for a complete presentation for the interfaces to the modules. This will provide the necessary detail for supporting both thorough testing of the TOE (using components from ATE_DPT), and the assessment of vulnerabilities.

In the context of the level of formality of the low-level design, informal, semiformal and formal are considered to be hierarchical in nature. Thus, ADV_LLD.1.1C may also be met with either a semiformal or formal low-level design, and ADV_LLD.2.1C may also be met with a formal low-level design.

ADV_LLD.1 Descriptive low-level design

Dependencies:

ADV_HLD.2 Security enforcing high-level design

ADV_RCR.1 Informal correspondence demonstration

Developer action elements:

ADV_LLD.1.1D **The developer shall provide the low-level design of the TSF.**

Content and presentation of evidence elements:

ADV_LLD.1.1C **The presentation of the low-level design shall be informal.**

ADV_LLD.1.2C The low-level design shall be internally consistent.

ADV_LLD.1.3C **The low-level design shall describe the TSF in terms of modules.**

ADV_LLD.1.4C **The low-level design shall describe the purpose of each module.**

ADV_LLD.1.5C **The low-level design shall define the interrelationships between the modules in terms of provided security functionality and dependencies on other modules.**

ADV_LLD.1.6C **The low-level design shall describe how each TSP-enforcing function is provided.**

ADV_LLD.1.7C The low-level design shall identify all interfaces to the modules of the TSF.

ADV_LLD.1.8C The low-level design shall identify which of the interfaces to the modules of the TSF are externally visible.

ADV_LLD.1.9C **The low-level design shall describe the purpose and method of use of all interfaces to the modules of the TSF, providing details of effects, exceptions and error messages, as appropriate.**

ADV_LLD.1.10C **The low-level design shall describe the separation of the TSF into TSP-enforcing and other modules.**

Evaluator action elements:

ADV_LLD.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_LLD.1.2E **The evaluator shall determine that the low-level design is an accurate and complete instantiation of the TOE security functional requirements.**

ADV_LLD.2 Semiformal low-level design

Dependencies:

ADV_HLD.3 Semiformal high-level design

ADV_RCR.2 Semiformal correspondence demonstration

Developer action elements:

ADV_LLD.2.1D The developer shall provide the low-level design of the TSF.

Content and presentation of evidence elements:

Content and presentation of evidence elements:

ADV_LLD.2.1C The presentation of the low-level design shall be **semiformal**.

ADV_LLD.2.2C The low-level design shall be internally consistent.

ADV_LLD.2.3C The low-level design shall describe the TSF in terms of modules.

ADV_LLD.2.4C The low-level design shall describe the purpose of each module.

ADV_LLD.2.5C The low-level design shall define the interrelationships between the modules in terms of provided security functionality and dependencies on other modules.

ADV_LLD.2.6C The low-level design shall describe how each TSP-enforcing function is provided.

ADV_LLD.2.7C The low-level design shall identify all interfaces to the modules of the TSF.

ADV_LLD.2.8C The low-level design shall identify which of the interfaces to the modules of the TSF are externally visible.

ADV_LLD.2.9C The low-level design shall describe the purpose and method of use of all interfaces to the modules of the TSF, providing **complete** details of **all** effects, exceptions and error messages.

ADV_LLD.2.10C The low-level design shall describe the separation of the TSF into TSP-enforcing and other modules.

Evaluator action elements:

ADV_LLD.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_LLD.2.2E The evaluator shall determine that the low-level design is an accurate and complete instantiation of the TOE security functional requirements.

ADV_LLD.3 Formal low-level design

Dependencies:

ADV_HLD.5 Formal high-level design

ADV_RCR.3 Formal correspondence demonstration

Developer action elements:

ADV_LLD.3.1D The developer shall provide the low-level design of the TSF.

Content and presentation of evidence elements:

ADV_LLD.3.1C The presentation of the low-level design shall be **formal**.

ADV_LLD.3.2C The low-level design shall be internally consistent.

ADV_LLD.3.3C The low-level design shall describe the TSF in terms of modules.

ADV_LLD.3.4C The low-level design shall describe the purpose of each module.

ADV_LLD.3.5C The low-level design shall define the interrelationships between the modules in terms of provided security functionality and dependencies on other modules.

ADV_LLD.3.6C The low-level design shall describe how each TSP-enforcing function is provided.

ADV_LLD.3.7C The low-level design shall identify all interfaces to the modules of the TSF.

ADV_LLD.3.8C The low-level design shall identify which of the interfaces to the modules of the TSF are externally visible.

ADV_LLD.3.9C The low-level design shall describe the purpose and method of use of all interfaces to the modules of the TSF, providing complete details of all effects, exceptions and error messages.

ADV_LLD.3.10C The low-level design shall describe the separation of the TSF into TSP-enforcing and other modules.

Evaluator action elements:

ADV_LLD.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_LLD.3.2E The evaluator shall determine that the low-level design is an accurate and complete instantiation of the TOE security functional requirements.

10.6 Representation correspondence (ADV_RCR)

Objectives

The correspondence between the various TSF representations (i.e. TOE summary specification, functional specification, high-level design, low-level design, implementation representation) addresses the correct and complete instantiation of the requirements to the least abstract TSF representation provided. This conclusion is achieved by step-wise refinement and the cumulative results of correspondence determinations between all adjacent abstractions of representation.

Component levelling

The components in this family are levelled on the basis of the level of rigour of the dependent TSF representations, and thus reflect the level of rigour that can be obtained in the correspondence between the various abstractions of TSF representation.

Application notes

The developer must demonstrate to the evaluator that the most detailed, or least abstract, TSF representation provided is an accurate, consistent, and complete instantiation of the functions expressed as functional requirements in the ST. This is accomplished by showing correspondence between adjacent representations at a commensurate level of rigour.

This family of requirements is not intended to address correspondence relating to the TSP model or the TSP. Rather, as shown in Figure 10.2, it is intended to address correspondence between various TSF representations (i.e. the TOE summary specification, functional specification, high-level design, low-level design, and implementation representation) that are provided.

The ADV_RCR.*.1C elements refer to “all relevant security functionality” in defining the scope of what must be refined between an adjacent pair of TSF representations. For the refinements between the TOE summary specification and the functional specification, this element requires only that the TOE security functions in the TOE summary specification be refined in the functional specification, and does not require that the functional specification contain any details regarding assurance measures (which are presented in the TOE summary specification). Where the implementation representation is only provided for a subset of the TSF (as in ADV_IMP.1), the required refinements between the low-level design and the implementation representation are limited to the security functionality that is presented in the implementation representation. In all other cases, this element requires that all parts of the more abstract TSF representation be refined in the less abstract TSF representation.

In the context of the level of formality for correspondence between adjacent TSF representations, informal, semiformal and formal are considered to be hierarchical in nature. Thus, ADV_RCR.2.2C and ADV_RCR.3.2C may be met with a formal proof of correspondence, and in the absence of any requirements on its level of formality, a demonstration of correspondence may be informal, semiformal or formal.

ADV_RCR.1 Informal correspondence demonstration

Dependencies:

No dependencies.

Developer action elements:

ADV_RCR.1.1D **The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.**

Content and presentation of evidence elements:

ADV_RCR.1.1C **For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.**

Evaluator action elements:

ADV_RCR.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ADV_RCR.2 Semiformal correspondence demonstration

Dependencies:

No dependencies.

Developer action elements:

ADV_RCR.2.1D The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

Content and presentation of evidence elements:

ADV_RCR.2.1C For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

ADV_RCR.2.2C For each adjacent pair of provided TSF representations, **where portions of both representations are at least semiformally specified,** the demonstration of correspondence between **those portions of** the representations **shall** be **semiformal**.

Evaluator action elements:

ADV_RCR.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_RCR.3 Formal correspondence demonstration

Application notes

The developer must either demonstrate or prove correspondence, as described in the requirements below, commensurate with the level of rigour of presentation style. For example, correspondence must be proven when corresponding representations are formally specified.

Dependencies:

No dependencies.

Developer action elements:

ADV_RCR.3.1D The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

ADV_RCR.3.2D **For those corresponding portions of representations that are formally specified, the developer shall prove that correspondence.**

Content and presentation of evidence elements:

ADV_RCR.3.1C For each adjacent pair of provided TSF representations, the analysis shall **prove or** demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

ADV_RCR.3.2C For each adjacent pair of provided TSF representations, where portions of **one** representation are **semiformally specified and the other** at least semiformally specified, the demonstration of correspondence between those portions of the representations shall be semiformal.

ADV_RCR.3.3C **For each adjacent pair of provided TSF representations, where portions of both representations are formally specified, the proof of correspondence between those portions of the representations shall be formal.**

Evaluator action elements:

ADV_RCR.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_RCR.3.2E **The evaluator shall determine the accuracy of the proofs of correspondence by selectively verifying the formal analysis.**

10.7 Security policy modeling (ADV_SPM)

Objectives

It is the objective of this family to provide additional assurance that the security functions in the functional specification enforce the policies in the TSP. This is accomplished via the development of a security policy model that is based on a subset of the policies of the TSP, and establishing a correspondence between the functional specification, the security policy model, and these policies of the TSP.

Component levelling

The components in this family are levelled on the basis of the degree of formality required of the TSP model, and the degree of formality required of the correspondence between the TSP model and the functional specification.

Application notes

While a TSP may include any policies, TSP models have traditionally represented only subsets of those policies, because modeling certain policies is currently beyond the state of the art. The current state of the art determines the policies that can be modeled, and the PP/ST author should identify specific functions and associated policies that can, and thus are required to be, modeled. At the very least, access control and information flow control policies are required to be modeled (if they are part of the TSP) since they are currently within the state of the art.

For each of the components within this family, there is a requirement to describe the rules and characteristics of applicable policies of the TSP in the TSP model and to ensure that the TSP model satisfies the corresponding policies of the TSP. The “rules” and “characteristics” of a TSP model are intended to allow flexibility in the type of model that may be developed (e.g. state transition, non-interference). For example, rules may be represented as “properties” (e.g. simple security property) and characteristics may be represented as definitions such as “initial state”, “secure state”, “subjects” and “objects”.

In the context of the level of formality of the TSP model and the correspondence between the TSP model and the functional specification, informal, semiformal and formal are considered to be hierarchical in nature. Thus, ADV_SPM.1.1C may also be met with either a semiformal or formal TSP model, and ADV_SPM.2.1C may also be met with a formal TSP model. Furthermore, ADV_SPM.2.5C and ADV_SPM.3.5C may be met with a formal proof of correspondence. Finally, in the absence of any requirements on its level of formality, a demonstration of correspondence may be informal, semiformal or formal.

ADV_SPM.1 Informal TOE security policy model

Dependencies:

ADV_FSP.1 Informal functional specification

Developer action elements:

ADV_SPM.1.1D The developer shall provide a TSP model.

ADV_SPM.1.2D The developer shall demonstrate correspondence between the functional specification and the TSP model.

Content and presentation of evidence elements:

ADV_SPM.1.1C The TSP model shall be informal.

ADV_SPM.1.2C The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_SPM.1.3C The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modeled.

ADV_SPM.1.4C The demonstration of correspondence between the TSP model and the functional specification shall show that all of the security functions in the functional specification are consistent and complete with respect to the TSP model.

Evaluator action elements:

ADV_SPM.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_SPM.2 Semiformal TOE security policy model

Dependencies:

ADV_FSP.1 Informal functional specification

Developer action elements:

ADV_SPM.2.1D The developer shall provide a TSP model.

ADV_SPM.2.2D The developer shall demonstrate correspondence between the functional specification and the TSP model.

Content and presentation of evidence elements:

ADV_SPM.2.1C The TSP model shall be **semiformal**.

ADV_SPM.2.2C The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_SPM.2.3C The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modeled.

ADV_SPM.2.4C The demonstration of correspondence between the TSP model and the functional specification shall show that all of the security functions in the functional specification are consistent and complete with respect to the TSP model.

ADV_SPM.2.5C Where the functional specification is at least semiformal, the demonstration of correspondence between the TSP model and the functional specification shall be semiformal.

Evaluator action elements:

ADV_SPM.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_SPM.3 Formal TOE security policy model

Dependencies:

ADV_FSP.1 Informal functional specification

Developer action elements:

ADV_SPM.3.1D The developer shall provide a TSP model.

ADV_SPM.3.2D The developer shall demonstrate **or prove, as appropriate,** correspondence between the functional specification and the TSP model.

Content and presentation of evidence elements:

ADV_SPM.3.1C The TSP model shall be **formal**.

ADV_SPM.3.2C The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_SPM.3.3C The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modeled.

ADV_SPM.3.4C The demonstration of correspondence between the TSP model and the functional specification shall show that all of the security functions in the functional specification are consistent and complete with respect to the TSP model.

ADV_SPM.3.5C Where the functional specification is **semiformal**, the demonstration of correspondence between the TSP model and the functional specification shall be semiformal.

ADV_SPM.3.6C Where the functional specification is formal, the proof of correspondence between the TSP model and the functional specification shall be formal.

DRAFT

Evaluator action elements:

ADV_SPM.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

11 Class AGD: Guidance documents

The guidance documents class provides the requirements for user and administrator guidance documentation. For the secure administration and use of the TOE it is necessary to describe all relevant aspects for the secure application of the TOE.

Figure 11.1 shows the families within this class, and the hierarchy of components within the families.

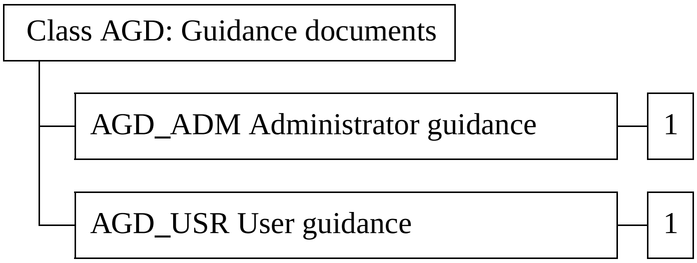

Figure 11.1 - Guidance documents class decomposition

11.1 Administrator guidance (AGD_ADM)

Objectives

Administrator guidance refers to written material that is intended to be used by those persons responsible for configuring, maintaining, and administering the TOE in a correct manner for maximum security. Because the secure operation of the TOE is dependent upon the correct performance of the TSF, persons responsible for performing these functions are trusted by the TSF. Administrator guidance is intended to help administrators understand the security functions provided by the TOE, including both those functions that require the administrator to perform security-critical actions and those functions that provide security-critical information.

Component levelling

This family contains only one component.

Application notes

The requirements AGD_ADM.1.3C and AGD_ADM.1.7C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the administrator guidance.

The concept of secure values, as employed in AGD_ADM.1.5C, has relevance where an administrator has control over security parameters. Guidance needs to be provided on secure and insecure settings for such parameters. This concept is related to the use of the Part 2 component FMT_MSA.2.

AGD_ADM.1 Administrator guidance

Dependencies:

ADV_FSP.1 Informal functional specification

Developer action elements:

AGD_ADM.1.1D **The developer shall provide administrator guidance addressed to system administrative personnel.**

Content and presentation of evidence elements:

AGD_ADM.1.1C **The administrator guidance shall describe the administrative functions and interfaces available to the administrator of the TOE.**

AGD_ADM.1.2C **The administrator guidance shall describe how to administer the TOE in a secure manner.**

AGD_ADM.1.3C **The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.**

AGD_ADM.1.4C **The administrator guidance shall describe all assumptions regarding user behaviour that are relevant to secure operation of the TOE.**

AGD_ADM.1.5C **The administrator guidance shall describe all security parameters under the control of the administrator, indicating secure values as appropriate.**

AGD_ADM.1.6C **The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.**

AGD_ADM.1.7C **The administrator guidance shall be consistent with all other documents supplied for evaluation.**

AGD_ADM.1.8C **The administrator guidance shall describe all security requirements on the IT environment that are relevant to the administrator.**

Evaluator action elements:

AGD_ADM.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

11.2 User guidance (AGD_USR)

Objectives

User guidance refers to material that is intended to be used by non-administrative human users of the TOE, and by others (e.g. programmers) using the TOE's external interfaces. User guidance describes the security functions provided by the TSF and provides instructions and guidelines, including warnings, for its secure use.

The user guidance provides a basis for assumptions about the use of the TOE and a measure of confidence that non-malicious users, application providers and others exercising the external interfaces of the TOE will understand the secure operation of the TOE and will use it as intended.

Component levelling

This family contains only one component.

Application notes

The requirements AGD_USR.1.3.C and AGD_USR.1.5C encompass the aspect that any warnings to the users of a TOE with regard to the TOE security environment and the security objectives described in the PP/ST are appropriately covered in the user guidance.

In many cases it may be appropriate that guidance is provided in separate documents: one for human users, and one for application programmers and/or hard-ware designers using software or hardware interfaces.

AGD_USR.1 User guidance

Dependencies:

ADV_FSP.1 Informal functional specification

Developer action elements:

AGD_USR.1.1D **The developer shall provide user guidance.**

Content and presentation of evidence elements:

AGD_USR.1.1C **The user guidance shall describe the functions and interfaces available to the non-administrative users of the TOE.**

AGD_USR.1.2C **The user guidance shall describe the use of user-accessible security functions provided by the TOE.**

AGD_USR.1.3C **The user guidance shall contain warnings about user-accessible functions and privileges that should be controlled in a secure processing environment.**

AGD_USR.1.4C **The user guidance shall clearly present all user responsibilities necessary for secure operation of the TOE, including those related to assumptions regarding user behaviour found in the statement of TOE security environment.**

AGD_USR.1.5C **The user guidance shall be consistent with all other documentation supplied for evaluation.**

AGD_USR.1.6C **The user guidance shall describe all security requirements on the IT environment that are relevant to the user.**

Evaluator action elements:

AGD_USR.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

12 Class ALC: Life cycle support

Life-cycle support is an aspect of establishing discipline and control in the processes of refinement of the TOE during its development and maintenance. Confidence in the correspondence between the TOE security requirements and the TOE is greater if security analysis and the production of the evidence are done on a regular basis as an integral part of the development and maintenance activities.

Figure 12.1 shows the families within this class, and the hierarchy of components within the families.

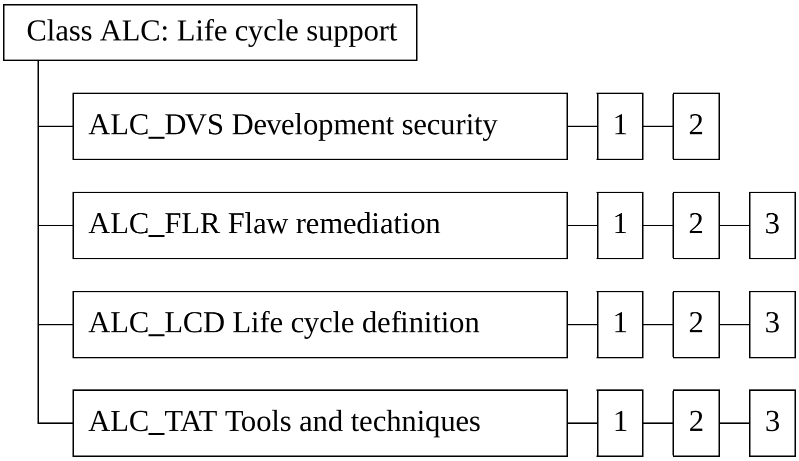
Class ALC: Life cycle support
ALC_DVS Development security
1
2
ALC_FLR Flaw remediation
1
2
3
ALC_LCD Life cycle definition
1
2
3
ALC_TAT Tools and techniques
1
2
3

Figure 12.1 -Life-cycle support class decomposition

12.1 Development security (ALC_DVS)

Objectives

Development security is concerned with physical, procedural, personnel, and other security measures that may be used in the development environment to protect the TOE. It includes the physical security of the development location and any procedures used to select development staff.

Component levelling

The components in this family are levelled on the basis of whether justification of the sufficiency of the security measures is required.

Application notes

This family deals with measures to remove or reduce threats existing at the developer’s site. Conversely, threats to be countered at the TOE user’s site are normally covered in the security environment section of a PP or ST.

The evaluator should determine whether there is a need for visiting the developer’s site in order to confirm that the requirements of this family are met.

It is recognised that confidentiality may not always be an issue for the protection of the TOE in its development environment. The use of the word “necessary” allows for the selection of appropriate safeguards.

ALC_DVS.1 Identification of security measures

Dependencies:

No dependencies.

Developer action elements:

ALC_DVS.1.1D **The developer shall produce development security documentation.**

Content and presentation of evidence elements:

ALC_DVS.1.1C **The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.**

ALC_DVS.1.2C **The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.**

Evaluator action elements:

ALC_DVS.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ALC_DVS.1.2E **The evaluator shall confirm that the security measures are being applied.**

ALC_DVS.2 Sufficiency of security measures

Dependencies:

No dependencies.

Developer action elements:

ALC_DVS.2.1D The developer shall produce development security documentation.

Content and presentation of evidence elements:

ALC_DVS.2.1C The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.

ALC_DVS.2.2C The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

ALC_DVS.2.3C **The evidence shall justify that the security measures provide the necessary level of protection to maintain the confidentiality and integrity of the TOE.**

Evaluator action elements:

ALC_DVS.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_DVS.2.2E The evaluator shall confirm that the security measures are being applied.

12.2 Flaw remediation (ALC_FLR)

Objectives

Flaw remediation requires that discovered flaws be tracked and corrected by the developer. Although future compliance with flaw remediation procedures cannot be determined at the time of the TOE evaluation, it is possible to evaluate the policies and procedures that a developer has in place to track and correct flaws, and to distribute the flaw information and corrections.

Component levelling

The components in this family are levelled on the basis of the increasing extent in scope of the flaw remediation procedures and the rigour of the flaw remediation policies.

Application notes

This family provides assurance that the TOE will be maintained and supported in the future; requiring the TOE developer to track and correct flaws in the TOE. Specifically that security flaws will be properly corrected and correction will be distributed. However, this family does not impose requirements beyond the current evaluation.

The flaw remediation procedures should describe the methods for dealing with all types of flaws encountered. Some flaws may not be fixable immediately. There may be some occasions where a flaw cannot be fixed and other (e.g. procedural) measures must be taken. The documentation provided should cover the procedures for providing the operational sites with fixes, and informing flaws found where fixes are delayed (and what to do in the mean time) or when fixes are not possible.

ALC_FLR.1 Basic flaw remediation

Dependencies:

No dependencies.

Developer action elements:

ALC_FLR.1.1D **The developer shall document the flaw remediation procedures.**

Content and presentation of evidence elements:

ALC_FLR.1.1C **The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.**

ALC_FLR.1.2C **The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.**

ALC_FLR.1.3C **The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.**

ALC_FLR.1.4C **The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.**

Evaluator action elements:

ALC_FLR.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ALC_FLR.2 Flaw reporting procedures

Dependencies:

No dependencies.

Developer action elements:

ALC_FLR.2.1D The developer shall document the flaw remediation procedures.

ALC_FLR.2.2D **The developer shall establish a procedure for accepting and acting upon user reports of security flaws and requests for corrections to those flaws.**

Content and presentation of evidence elements:

ALC_FLR.2.1C The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

ALC_FLR.2.2C The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.

ALC_FLR.2.3C The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.

ALC_FLR.2.4C The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

ALC_FLR.2.5C **The procedures for processing reported security flaws shall ensure that any reported flaws are corrected and the correction issued to TOE users.**

ALC_FLR.2.6C **The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.**

Evaluator action elements:

ALC_FLR.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_FLR.3 Systematic flaw remediation

Dependencies:

No dependencies.

Developer action elements:

ALC_FLR.3.1D The developer shall document the flaw remediation procedures.

ALC_FLR.3.2D The developer shall establish a procedure for accepting and acting upon user reports of security flaws and requests for corrections to those flaws.

ALC_FLR.3.3D **The developer shall designate one or more specific points of contact for user reports and inquiries about security issues involving the TOE.**

Content and presentation of evidence elements:

ALC_FLR.3.1C The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

ALC_FLR.3.2C The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.

ALC_FLR.3.3C The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.

ALC_FLR.3.4C The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

ALC_FLR.3.5C The procedures for processing reported security flaws shall ensure that any reported flaws are corrected and the correction issued to TOE users.

ALC_FLR.3.6C The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.

ALC_FLR.3.7C **The flaw remediation procedures shall include a procedure requiring timely responses for the automatic distribution of security flaw reports and the associated corrections to registered users who might be affected by the security flaw.**

Evaluator action elements:

ALC_FLR.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

12.3 Life cycle definition(ALC_LCD)

Objectives

Poorly controlled development and maintenance of the TOE can result in a flawed implementation of a TOE (or a TOE that does not meet all of its security requirements). This, in turn, results in security violations. Therefore, it is important that a model for the development and maintenance of a TOE be established as early as possible in the TOE's life-cycle.

Using a model for the development and maintenance of a TOE does not guarantee that the TOE will be free of flaws, nor does it guarantee that the TOE will meet all of its security functional requirements. It is possible that the model chosen will be insufficient or inadequate and therefore no benefits in the quality of the TOE could be observed. Using a life-cycle model that has been approved by some group of experts (e.g. academic experts, standards bodies) improves the chances that the development and maintenance models will contribute to the overall quality of the TOE.

Component levelling

The components in this family are levelled on the basis of increasing requirements for standardisation and measurability of the life-cycle model, and for compliance with that model.

Application notes

A life-cycle model encompasses the procedures, tools and techniques used to develop and maintain the TOE. Aspects of the process that may be covered by such a model include design methods, review procedures, project management controls, change control procedures, test methods and acceptance procedures. An effective life-cycle model will address these aspects of the development and maintenance process within an overall management structure that assigns responsibilities and monitors progress.

Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming relevant after the completion of the evaluation, its evaluation adds assurance through an analysis of the life-cycle information for the TOE provided at the time of the evaluation.

A standardised life-cycle model is a model that has been approved by some group of experts (e.g. academic experts, standards bodies).

A measurable life-cycle model is a model with arithmetic parameters and/or metrics that measure TOE development properties (e.g. source code complexity metrics).

A life-cycle model provides for the necessary control over the development and maintenance of the TOE, if the developer can supply information that shows that the model appropriately minimises the danger of security violations in the TOE. Information given in the ST about the intended environment of the TOE and about the TOE's security objectives may be useful in defining the model for the portion of the life-cycle after the delivery of the TOE.

ALC_LCD.1 Developer defined life-cycle model

Dependencies:

No dependencies.

Developer action elements:

ALC_LCD.1.1D **The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.**

ALC_LCD.1.2D **The developer shall provide life-cycle definition documentation.**

Content and presentation of evidence elements:

ALC_LCD.1.1C **The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.**

ALC_LCD.1.2C **The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.**

Evaluator action elements:

ALC_LCD.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ALC_LCD.2 Standardised life-cycle model

Dependencies:

No dependencies.

Developer action elements:

ALC_LCD.2.1D The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.

ALC_LCD.2.2D The developer shall provide life-cycle definition documentation.

ALC_LCD.2.3D **The developer shall use a standardised life-cycle model to develop and maintain the TOE.**

Content and presentation of evidence elements:

ALC_LCD.2.1C The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.

ALC_LCD.2.2C The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.

ALC_LCD.2.3C **The life-cycle definition documentation shall explain why the model was chosen.**

ALC_LCD.2.4C **The life-cycle definition documentation shall explain how the model is used to develop and maintain the TOE**

ALC_LCD.2.5C **The life-cycle definition documentation shall demonstrate compliance with the standardised life-cycle model.**

Evaluator action elements:

ALC_LCD.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_LCD.3 Measurable life-cycle model

Dependencies:

No dependencies.

Developer action elements:

ALC_LCD.3.1D The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.

ALC_LCD.3.2D The developer shall provide life-cycle definition documentation.

ALC_LCD.3.3D The developer shall use a standardised **and measurable** life-cycle model to develop and maintain the TOE.

ALC_LCD.3.4D **The developer shall measure the TOE development using the standardised and measurable life-cycle model.**

Content and presentation of evidence elements:

ALC_LCD.3.1C The life-cycle definition documentation shall describe the model used to develop and maintain the TOE, **including the details of its arithmetic parameters and/or metrics used to measure the TOE development against the model**.

ALC_LCD.3.2C The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.

ALC_LCD.3.3C The life-cycle definition documentation shall explain why the model was chosen.

ALC_LCD.3.4C The life-cycle definition documentation shall explain how the model is used to develop and maintain the TOE.

ALC_LCD.3.5C The life-cycle definition documentation shall demonstrate compliance with the standardised **and measurable** life-cycle model.

ALC_LCD.3.6C **The life-cycle documentation shall provide the results of the measurements of the TOE development using the standardised and measurable life-cycle model.**

Evaluator action elements:

ALC_LCD.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

12.4 Tools and techniques (ALC_TAT)

Objectives

Tools and techniques is an aspect of selecting tools that are used to develop, analyse and implement the TOE. It includes requirements to prevent ill-defined, inconsistent or incorrect development tools from being used to develop the TOE. This includes, but is not limited to, programming languages, documentation, implementation standards, and other parts of the TOE such as supporting runtime libraries.

Component levelling

The components in this family are levelled on the basis of increasing requirements on the description and scope of the implementation standards and the documentation of implementation dependent options.

Application notes

There is a requirement for well-defined development tools. These are tools that have been shown to be applicable without the need for intensive further clarification. For example, programming languages and computer aided design (CAD) systems that are based on an a standard published by standards bodies are considered to be well-defined.

Tools and techniques distinguishes between the implementation standards applied by the developer (ALC_TAT.2.3D) and the implementation standards for “all parts of the TOE” (ALC_TAT.3.3D) that additionally includes third party software, hardware, or firmware.

The requirement in ALC_TAT.1.2C is especially applicable to programming languages so as to ensure that all statements in the source code have an unambiguous meaning.

ALC_TAT.1 Well-defined development tools

Dependencies:

ADV_IMP.1 Subset of the implementation of the TSF

Developer action elements:

ALC_TAT.1.1D **The developer shall identify the development tools being used for the TOE.**

ALC_TAT.1.2D **The developer shall document the selected implementation dependent options of the development tools.**

Content and presentation of evidence elements:

ALC_TAT.1.1C **All development tools used for implementation shall be well-defined.**

ALC_TAT.1.2C **The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.**

ALC_TAT.1.3C **The documentation of the development tools shall unambiguously define the meaning of all implementation dependent options.**

Evaluator action elements:

ALC_TAT.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ALC_TAT.2 Compliance with implementation standards

Dependencies:

ADV_IMP.1 Subset of the implementation of the TSF

Developer action elements:

ALC_TAT.2.1D The developer shall identify the development tools being used for the TOE.

ALC_TAT.2.2D The developer shall document the selected implementation dependent options of the development tools.

ALC_TAT.2.3D **The developer shall describe the implementation standards to be applied.**

Content and presentation of evidence elements:

ALC_TAT.2.1C All development tools used for implementation shall be well-defined.

ALC_TAT.2.2C The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.

ALC_TAT.2.3C The documentation of the development tools shall unambiguously define the meaning of all implementation dependent options.

Evaluator action elements:

ALC_TAT.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_TAT.2.2E **The evaluator shall confirm that the implementation standards have been applied.**

ALC_TAT.3 Compliance with implementation standards - all parts

Dependencies:

ADV_IMP.1 Subset of the implementation of the TSF

Developer action elements:

ALC_TAT.3.1D The developer shall identify the development tools being used for the TOE.

ALC_TAT.3.2D The developer shall document the selected implementation dependent options of the development tools.

ALC_TAT.3.3D The developer shall describe the implementation standards **for all parts of the TOE**.

Content and presentation of evidence elements:

ALC_TAT.3.1C All development tools used for implementation shall be well-defined.

ALC_TAT.3.2C The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.

ALC_TAT.3.3C The documentation of the development tools shall unambiguously define the meaning of all implementation dependent options.

Evaluator action elements:

ALC_TAT.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_TAT.3.2E The evaluator shall confirm that the implementation standards have been applied.

13 Class ATE: Tests

The class “Tests” encompasses four families: coverage (ATE_COV), depth (ATE_DPT), independent testing (e.g. functional testing performed by evaluators) (ATE_IND), and functional tests (ATE_FUN). Testing helps to establish that the TOE security functional requirements are met. Testing provides assurance that the TOE satisfies at least the TOE security functional requirements, although it cannot establish that the TOE does no more than what was specified. Testing may also be directed toward the internal structure of the TSF, such as the testing of subsystems and modules against their specifications.

The aspects of coverage and depth have been separated from functional tests for reasons of increased flexibility in applying the components of the families. However, the requirements in these three families are intended to be applied together.

The independent testing family has dependencies on the other families to provide the necessary information to support the requirements, but is primarily concerned with independent evaluator actions.

The emphasis in this class is on confirmation that the TSF operates according to its specification. This will include both positive testing based on functional requirements, and negative testing to check that undesirable behaviour is absent. This class does not address penetration testing, which is directed toward finding vulnerabilities that enable a user to violate the security policy. Penetration testing is based upon an analysis of the TOE that specifically seeks to identify vulnerabilities in the design and implementation of the TSF, and is addressed separately as an aspect of vulnerability assessment in the class AVA.

Figure 13.1 shows the families within this class, and the hierarchy of components within the families.

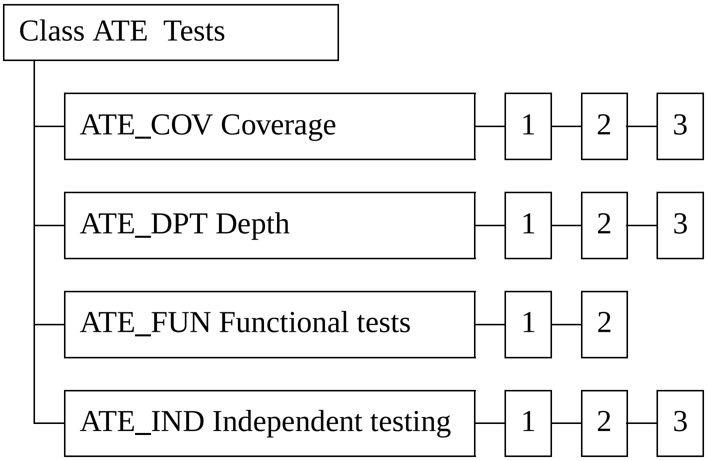

Figure 13.1 -Tests class decomposition

13.1 Coverage (ATE_COV)

Objectives

This family addresses those aspects of testing that deal with completeness of test coverage. That is, it addresses the extent to which the TSF is tested, and whether or not the testing is sufficiently extensive to demonstrate that the TSF operates as specified.

Component levelling

The components in this family are levelled on the basis of increasing rigour of interface testing, and increasing rigour of the analysis of the sufficiency of the tests to demonstrate that the TSF operates in accordance with its functional specification.

Application notes

In this component the developer is required to show how the tests that have been identified correspond to the TSF as described in the functional specification. This can be achieved by a statement of correspondence, perhaps using a table. This information is required to support the evaluator in planning the test programme for the evaluation. At this level there is no requirement for complete coverage of every aspect of the TSF by the developer, and the evaluator will need to take account of any deficiencies in this area.

ATE_COV.1 Evidence of coverage

Objectives

In this component, the objective is to establish that the TSF has been tested against its functional specification. This is to be achieved through an examination of developer evidence of correspondence.

Application notes

While the testing objective is to cover the TSF, there is no requirement to provide anything to verify this assertion other than an informal mapping of tests to the functional specification and the testing data itself.

Dependencies:

ADV_FSP.1 Informal functional specification

ATE_FUN.1 Functional testing

Developer action elements:

ATE_COV.1.1D **The developer shall provide evidence of the test coverage.**

Content and presentation of evidence elements:

ATE_COV.1.1C **The evidence of the test coverage shall show the correspondence between the tests identified in the test documentation and the TSF as described in the functional specification.**

Evaluator action elements:

ATE_COV.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ATE_COV.2 Analysis of coverage

Objectives

In this component, the objective is to establish that the TSF has been tested against its functional specification in a systematic manner. This is to be achieved through an examination of developer analysis of correspondence.

Application notes

The developer is required to demonstrate that the tests which have been identified include testing of all of the security functions as described in the functional specification. The analysis should not only show the correspondence between tests and security functions, but should provide also sufficient information for the evaluator to determine how the functions have been exercised. This information can be used in planning for additional evaluator tests. Although at this level the developer has to demonstrate that each of the functions within the functional specification has been tested, the amount of testing of each function need not be exhaustive.

Dependencies:

ADV_FSP.1 Informal functional specification

ATE_FUN.1 Functional testing

Developer action elements:

ATE_COV.2.1D The developer shall provide **an analysis** of the test coverage.

Content and presentation of evidence elements:

ATE_COV.2.1C The **analysis** of the test coverage shall **demonstrate** the correspondence between the tests identified in the test documentation and the TSF as described in the functional specification.

ATE_COV.2.2C **The analysis of the test coverage shall demonstrate that the correspondence between the TSF as described in the functional specification and the tests identified in the test documentation is complete.**

Evaluator action elements:

ATE_COV.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_COV.3 Rigorous analysis of coverage

Objectives

In this component, the objective is to establish that the TSF has been tested against its functional specification in a systematic and exhaustive manner. This is to be achieved through an examination of developer analysis of correspondence.

Application notes

The developer is required to provide a convincing argument that the tests which have been identified cover all security functions, and that the testing of each security function is complete. There will remain little scope for the evaluator to devise additional functional tests of the TSF interfaces based on the functional specification, as they will have been exhaustively tested. Nevertheless, the evaluator should strive to devise such tests.

Dependencies:

ADV_FSP.1 Informal functional specification

ATE_FUN.1 Functional testing

Developer action elements:

ATE_COV.3.1D The developer shall provide an analysis of the test coverage.

Content and presentation of evidence elements:

ATE_COV.3.1C The analysis of the test coverage shall show the correspondence between the tests identified in the test documentation and the TSF as described in the functional specification.

ATE_COV.3.2C The analysis of the test coverage shall demonstrate that the correspondence between the TSF as described in the functional specification and the tests identified in the test documentation is complete.

ATE_COV.3.3C The analysis of the test coverage shall rigorously demonstrate that all external interfaces of the TSF identified in the functional specification have been completely tested.

Evaluator action elements:

ATE_COV.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

13.2 Depth (ATE_DPT)

Objectives

The components in this family deal with the level of detail to which the TSF is tested. Testing of security functions is based upon increasing depth of information derived from analysis of the representations.

The objective is to counter the risk of missing an error in the development of the TOE. Additionally, the components of this family, especially as testing is more concerned with the internal structure of the TSF, are more likely to discover any malicious code that has been inserted.

Testing that exercises specific internal interfaces can provide assurance not only that the TSF exhibits the desired external security behaviour, but also that this behaviour stems from correctly operating internal mechanisms.

Component levelling

The components in this family are levelled on the increasing level of detail provided in the TSF representations, from the high-level design to the implementation representation. This levelling reflects the representations presented in the ADV class.

Application notes

The specific amount and type of documentation and evidence will, in general, be determined by the chosen component from ATE_FUN.

Testing at the level of the functional specification is addressed by ATE_COV.

The principle adopted within this family is that the level of testing be appropriate to the level of assurance being sought. Where higher components are applied, the test results will need to demonstrate that the implementation of the TSF is consistent with its design. For example, the HLD should describe each of the subsystems and also describe the interfaces between these subsystems in sufficient detail. Evidence of testing must show that the internal interfaces between subsystems have been exercised. This may be achieved through testing via the external interfaces of the TSF, or by testing of the subsystem interfaces in isolation, perhaps employing a test harness. In cases where some aspects of an internal interface cannot be tested via the external interfaces there should either be justification that these aspects need not be tested, or the internal interface needs to be tested directly. In the latter case the high-level design needs to be sufficiently detailed in order to facilitate direct testing. The higher components in this family aim to check the correct operation of internal interfaces that become visible as the design becomes less abstract. When these components are applied it will be more difficult to provide adequate evidence of the depth of testing using the TSF’s external interfaces alone, and modular testing will usually be necessary.

ATE_DPT.1 Testing - high level design

Objectives

The subsystems of a TSF provide a high level description of the internal workings of the TSF. Testing at the level of the subsystems, in order to demonstrate the presence of any flaws, provides assurance that the TSF subsystems have been correctly realised.

Application notes

The developer is expected to describe the testing of the high level design of the TSF in terms of “subsystems”. The term “subsystem” is used to express the notion of decomposing the TSF into a relatively small number of parts.

Dependencies:

ADV_HLD.1 Descriptive high-level design

ATE_FUN.1 Functional testing

Developer action elements:

ATE_DPT.1.1D **The developer shall provide the analysis of the depth of testing.**

Content and presentation of evidence elements:

ATE_DPT.1.1C **The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with its high-level design.**

Evaluator action elements:

ATE_DPT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_DPT.2 Testing - low level design

Objectives

The subsystems of a TSF provide a high level description of the internal workings of the TSF. Testing at the level of the subsystems, in order to demonstrate the presence of any flaws, provides assurance that the TSF subsystems have been correctly realised.

The modules of a TSF provide a description of the internal workings of the TSF. Testing at the level of the modules, in order to demonstrate the presence of any flaws, provides assurance that the TSF modules have been correctly realised.

Application notes

The developer is expected to describe the testing of the high level design of the TSF in terms of “subsystems”. The term “subsystem” is used to express the notion of decomposing the TSF into a relatively small number of parts.

The developer is expected to describe the testing of the low level design of the TSF in terms of “modules”. The term “modules” is used to express the notion of decomposing each of the “subsystems” of the TSF into a relatively small number of parts.

Dependencies:

ADV_HLD.2 Security enforcing high-level design

ADV_LLD.1 Descriptive low-level design

ATE_FUN.1 Functional testing

Developer action elements:

ATE_DPT.2.1D The developer shall provide the analysis of the depth of testing.

Content and presentation of evidence elements:

ATE_DPT.2.1C The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with its high-level design **and low-level design**.

Evaluator action elements:

ATE_DPT.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_DPT.3 Testing - implementation

Objectives

The subsystems of a TSF provide a high level description of the internal workings of the TSF. Testing at the level of the subsystems, in order to demonstrate the presence of any flaws, provides assurance that the TSF subsystems have been correctly realised.

The modules of a TSF provide a description of the internal workings of the TSF. Testing at the level of the modules, in order to demonstrate the presence of any flaws, provides assurance that the TSF modules have been correctly realised.

The implementation representation of a TSF provides a detailed description of the internal workings of the TSF. Testing at the level of the implementation, in order to demonstrate the presence of any flaws, provides assurance that the TSF implementation has been correctly realised.

Application notes

The developer is expected to describe the testing of the high level design of the TSF in terms of “subsystems”. The term “subsystem” is used to express the notion of decomposing the TSF into a relatively small number of parts.

The developer is expected to describe the testing of the low level design of the TSF in terms of “modules”. The term “modules” is used to express the notion of decomposing each of the “subsystems” of the TSF into a relatively small number of parts.

The implementation representation is the one which is used to generate the TSF itself (e.g. source code which is then compiled).

Dependencies:

ADV_HLD.2 Security enforcing high-level design

ADV_IMP.2 Implementation of the TSF

ADV_LLD.1 Descriptive low-level design

ATE_FUN.1 Functional testing

Developer action elements:

ATE_DPT.3.1D The developer shall provide the analysis of the depth of testing.

Content and presentation of evidence elements:

ATE_DPT.3.1C The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with its high-level design, low-level design **and implementation representation**.

Evaluator action elements:

ATE_DPT.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

13.3 Functional tests (ATE_FUN)

Objectives

Functional testing performed by the developer establishes that the TSF exhibits the properties necessary to satisfy the functional requirements of its PP/ST. Such functional testing provides assurance that the TSF satisfies at least the security functional requirements, although it cannot establish that the TSF does no more than what was specified. The family "Functional tests" is focused on the type and amount of documentation or support tools required, and what is to be demonstrated through developer testing. Functional testing is not limited to positive confirmation that the required security functions are provided, but may also include negative testing to check for the absence of particular undesired behaviour (often based on the inversion of functional requirements).

This family contributes to providing assurance that the likelihood of undiscovered flaws is relatively small.

The families ATE_COV, ATE_DPT and ATE_FUN are used in combination to define the evidence of testing to be supplied by a developer. Independent functional testing by the evaluator is specified by ATE_IND.

This family contains two components, the higher requiring that ordering dependencies are analysed.

Application notes

Procedures for performing tests are expected to provide instructions for using test programs and test suites, including the test environment, test conditions, test data parameters and values. The test procedures should also show how the test results is derived from the test inputs.

This family specifies requirements for the presentation of all test plans, procedures and results. Thus the quantity of information that must be presented will vary in accordance with the use of ATE_COV and ATE_DPT.

ATE_FUN.1 Functional testing

Objectives

The objective is for the developer to demonstrate that all security functions perform as specified. The developer is required to perform testing and to provide test documentation.

Dependencies:

No dependencies.

Developer action elements:

ATE_FUN.1.1D **The developer shall test the TSF and document the results.**

ATE_FUN.1.2D **The developer shall provide test documentation.**

Content and presentation of evidence elements:

ATE_FUN.1.1C **The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.**

ATE_FUN.1.2C **The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.**

ATE_FUN.1.3C **The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function. These scenarios shall include any ordering dependencies on the results of other tests.**

ATE_FUN.1.4C **The expected test results shall show the anticipated outputs from a successful execution of the tests.**

ATE_FUN.1.5C **The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.**

Evaluator action elements:

ATE_FUN.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ATE_FUN.2 Ordered functional testing

Objectives

The objective is for the developer to demonstrate that all security functions perform as specified. The developer is required to perform testing and to provide test documentation.

In this component, an additional objective is to ensure that testing is structured such as to avoid circular arguments about the correctness of the portions of the TSF being tested.

Application notes

Ordering dependencies between tests can be of different forms. For example, test A provides a result to test B; test A cannot run before test B, since it breaks something required by test B; test failure in test B might be because of a failure in “untested” test A.

Although the test procedures may state pre-requisite initial test conditions in terms of ordering of tests, they may not provide a rationale for the ordering. An analysis of test ordering is an important factor in determining the adequacy of testing, as there is a possibility of faults being concealed by the ordering of tests.

Dependencies:

No dependencies.

Developer action elements:

ATE_FUN.2.1D The developer shall test the TSF and document the results.

ATE_FUN.2.2D The developer shall provide test documentation.

Content and presentation of evidence elements:

ATE_FUN.2.1C The test documentation shall consist of test plans, test procedure descriptions, expected test results and test results.

ATE_FUN.2.2C The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

ATE_FUN.2.3C The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function. These scenarios shall include any ordering dependencies on the results of other tests.

ATE_FUN.2.4C The expected test results shall show the anticipated outputs from a successful execution of the tests.

ATE_FUN.2.5C The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.

ATE_FUN.2.6C The test documentation shall include an analysis of the test procedure ordering dependencies.

Evaluator action elements:

ATE_FUN.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

13.4 Independent testing (ATE_IND)

Objectives

The objective is to demonstrate that the security functions perform as specified.

An additional objective is to counter the risk of an incorrect assessment of the test outcomes on the part of the developer that results in the incorrect implementation of the specifications, or overlooks code that is non-compliant with the specifications.

Component levelling

Levelling is based upon the amount of test documentation, test support and the amount of evaluator testing.

Application notes

The testing specified in this family can be supported by a party with specialised knowledge other than the evaluator (e.g. an independent laboratory, an objective consumer organisation). Testing requires an understanding of the TOE consistent with the performance of other assurance activities, and the evaluator retains responsibility for ensuring that the requirements of this family are properly addressed when such support it used.

This family deals with the degree to which there is independent functional testing of the TSF. Independent functional testing may take the form of repeating the developer’s functional tests, in whole or in part. It may also take the form of the augmentation of the developer’s functional tests, either to extend the scope or the depth of the developer’s tests, or to test for obvious public domain security weaknesses that could be applicable to the TOE. These activities are complementary, and an appropriate mix must be planned for each TOE, which takes into account the availability and coverage of test results, and the functional complexity of the TSF. A test plan should be developed that is consistent with the level of other assurance activities, and which, as greater assurance is required, includes larger samples of repeated tests, and more independent positive and negative functional tests by the evaluator.

Sampling of developer tests is intended to provide confirmation that the developer has carried out his planned test program on the TSF, and has correctly recorded the results. The size of sample selected will be influenced by the detail and quality of the developer’s functional test results. The evaluator will also need to consider the scope for devising additional tests, and the relative benefit that may be gained from effort in these two areas. It is recognised that repetition of all developer tests may be feasible and desirable in some cases, but may be very arduous and less productive in others. The highest component in this family should therefore be used with caution. Sampling will address the whole range of tests results available, including those supplied to meet the requirements of both ATE_COV and ATE_DPT.

There is also a need to consider the different configurations of the TOE that are included within the evaluation. The evaluator will need to assess the applicability of the results provided, and to plan his own testing accordingly.

Independent functional testing is distinct from penetration testing, the latter being based on an informed and systematic search for vulnerabilities in the design and/or implementation. Penetration testing is specified using the family AVA_VLA.

ATE_IND.1 Independent testing - conformance

Objectives

In this component, the objective is to demonstrate that the security functions perform as specified.

Application notes

The suitability of the TOE for testing is based on the access to the TOE, and the supporting documentation and information required (including any test software or tools) to run tests. The need for such support is addressed by the dependencies to other assurance families.

Additionally, suitability of the TOE for testing may be based on other considerations. For example, the version of the TOE submitted by the developer may not be the final version.

This component does not address the use of developer test results. It is applicable where such results are not available, and also in cases where the developer's testing is accepted without validation. The evaluator is required to devise and conduct tests with the objective of confirming that the TOE security functional requirements are met. The approach is to gain confidence in correct operation through representative testing, rather than to conduct every possible test. The extent of testing to be planned for this purpose is a methodology issue, and needs to be considered in the context of a particular TOE and the balance of other evaluation activities.

The reference to a subset of the TSF is intended to allow the evaluator to design an appropriate set of tests which is consistent with the objectives of the evaluation being conducted.

Dependencies:

ADV_FSP.1 Informal functional specification

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

ATE_IND.1.1D **The developer shall provide the TOE for testing.**

Content and presentation of evidence elements:

ATE_IND.1.1C **The TOE shall be suitable for testing.**

Evaluator action elements:

ATE_IND.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

ATE_IND.1.2E **The evaluator shall test a subset of the TSF as appropriate to confirm that the TOE operates as specified.**

ATE_IND.2 Independent testing - sample

Objectives

The objective is to demonstrate that the security functions perform as specified.

In this component, the objective is to support evaluator testing by selecting and repeating a sample of the developer testing.

Application notes

The suitability of the TOE for testing is based on the access to the TOE, and the supporting documentation and information required (including any test software or tools) to run tests. The need for such support is addressed by the dependencies to other assurance families.

Additionally, suitability of the TOE for testing may be based on other considerations (e.g. the version of the TOE submitted by the developer is not the final version).

The intent is that the developer should provide the evaluator with materials necessary for the efficient reproduction of developer tests. This may include such things as machine-readable test documentation, test programs, etc.

The developer is required to perform testing and to provide test documentation and test results. This is addressed by the ATE_FUN family.

This component contains a requirement that the evaluator has available test results from the developer to supplement the program of testing. The evaluator will repeat a sample of the developer's tests to gain confidence in the results obtained. Having established such confidence the evaluator will build upon the developer's testing by conducting additional tests that exercise the TOE in a different manner. By using a platform of validated developer test results the evaluator is able to gain confidence that the TOE operates correctly in a wider range of conditions than would be possible purely using the developer's own efforts, given a fixed level of resource. Having gained confidence that the developer has tested the TOE, the evaluator will also have more freedom, where appropriate, to concentrate testing in areas where examination of documentation or specialist knowledge has raised particular concerns.

Dependencies:

ADV_FSP.1 Informal functional specification

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

ATE_FUN.1 Functional testing

Developer action elements:

ATE_IND.2.1D The developer shall provide the TOE for testing.

Content and presentation of evidence elements:

ATE_IND.2.1C The TOE shall be suitable for testing.

ATE_IND.2.2C The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

Evaluator action elements:

ATE_IND.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.2.2E The evaluator shall test a subset of the TSF as appropriate to confirm that the TOE operates as specified.

ATE_IND.2.3E **The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.**

ATE_IND.3 Independent testing - complete

Objectives

The objective is to demonstrate that all security functions perform as specified.

In this component, the objective is to support evaluator testing by repeating all of the developer testing.

Application notes

The suitability of the TOE for testing is based on the access to the TOE, and the supporting documentation and information required (including any test software or tools) to run tests. The need for such support is addressed by the dependencies to other assurance families.

Additionally, suitability of the TOE for testing may be based on other considerations (e.g. the version of the TOE submitted by the developer is not the final version).

The developer is required to perform testing and to provide test documentation and test results. This is addressed by the ATE_FUN family.

In this component the evaluator must repeat all of the developer's tests as part of the programme of testing. As in the previous level the evaluator will also conduct tests that aim to exercise the TOE in a different manner from that achieved by the developer. In cases where developer testing has been exhaustive, there may remain little scope for this.

Dependencies:

ADV_FSP.1 Informal functional specification

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

ATE_FUN.1 Functional testing

Developer action elements:

ATE_IND.3.1D The developer shall provide the TOE for testing.

Content and presentation of evidence elements:

ATE_IND.3.1C The TOE shall be suitable for testing.

ATE_IND.3.2C The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

Evaluator action elements:

ATE_IND.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.3.2E The evaluator shall test a subset of the TSF as appropriate to confirm that the TOE operates as specified.

ATE_IND.3.3E The evaluator shall execute **all** tests in the test documentation to verify the developer test results.

14 Class AVA: Vulnerability assessment

The class addresses the existence of exploitable covert channels, the possibility of misuse or incorrect configuration of the TOE, the possibility to defeat probabilistic or permutational mechanisms, and the possibility of exploitable vulnerabilities introduced in the development or the operation of the TOE.

Figure 14.1 shows the families within this class, and the hierarchy of components within the families.

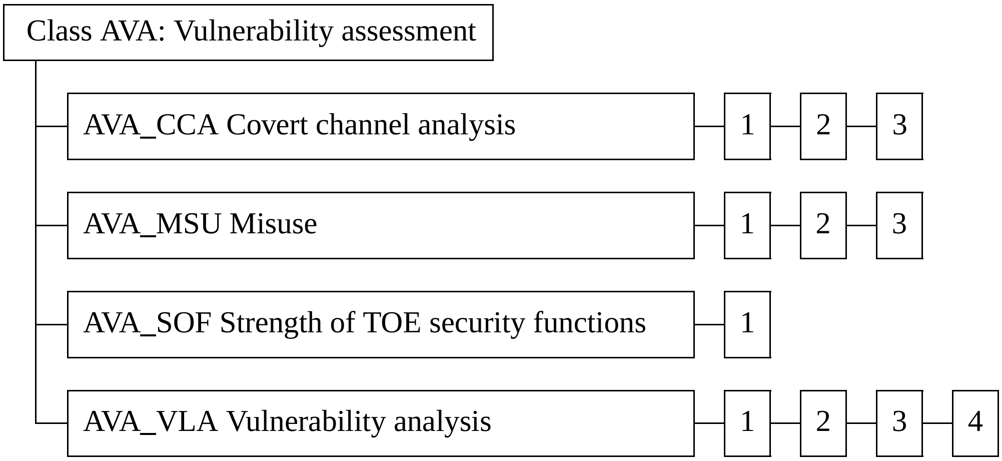
Class AVA: Vulnerability assessment
AVA_CCA Covert channel analysis
1
2
3
AVA_MSU Misuse
1
2
3
AVA_SOF Strength of TOE security functions
1
AVA_VLA Vulnerability analysis
1
2
3
4

Figure 14.1 -Vulnerability assessment class decomposition

14.1 Covert channel analysis (AVA_CCA)

Objectives

Covert channel analysis is carried out to determine the existence and potential capacity of unintended signalling channels (i.e. illicit information flows) that may be exploited.

The assurance requirements address the threat that unintended and exploitable signalling paths exist that may be exercised to violate the SFP.

Component levelling

The components are levelled on increasing rigour of covert channel analysis.

Application notes

Channel capacity estimations are based upon informal engineering measurements, as well as actual test measurements.

Examples of assumptions upon which the covert channel analysis is based may include processor speed, system or network configuration, memory size, and cache size.

The selective validation of the covert channel analysis through testing allows the evaluator the opportunity to verify any aspect of the covert channel analysis (e.g. identification, capacity estimation, elimination, monitoring, and exploitation scenarios). This does not impose a requirement to demonstrate the entire set of covert channel analysis results.

If there are no information flow control SFPs in the ST, this family of assurance requirements is no longer applicable, as this family applies only to information flow control SFPs.

AVA_CCA.1 Covert channel analysis

Objectives

The objective is to identify covert channels that are identifiable through analysis.

In this component, the objective is to perform informal search for covert channels.

Dependencies:

ADV_FSP.2 Fully defined external interfaces

ADV_IMP.2 Implementation of the TSF

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_CCA.1.1D **The developer shall conduct a search for covert channels for each information flow control policy.**

AVA_CCA.1.2D **The developer shall provide covert channel analysis documentation.**

Content and presentation of evidence elements:

AVA_CCA.1.1C **The analysis documentation shall identify covert channels and estimate their capacity.**

AVA_CCA.1.2C **The analysis documentation shall describe the procedures used for determining the existence of covert channels, and the information needed to carry out the covert channel analysis.**

AVA_CCA.1.3C **The analysis documentation shall describe all assumptions made during the covert channel analysis.**

AVA_CCA.1.4C **The analysis documentation shall describe the method used for estimating channel capacity, based on worst case scenarios.**

AVA_CCA.1.5C **The analysis documentation shall describe the worst case exploitation scenario for each identified covert channel.**

Evaluator action elements:

AVA_CCA.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

AVA_CCA.1.2E **The evaluator shall confirm that the results of the covert channels analysis show that the TOE meets its functional requirements.**

AVA_CCA.1.3E **The evaluator shall selectively validate the covert channel analysis through testing.**

AVA_CCA.2 Systematic covert channel analysis

Objectives

The objective is to identify covert channels that are identifiable through analysis.

In this component, the objective is to perform a systematic search for covert channels.

Application notes

Performing a covert channel analysis in a systematic way requires that the developer identify covert channels in a structured and repeatable way, as opposed to identifying covert channels in an ad-hoc fashion.

Dependencies:

ADV_FSP.2 Fully defined external interfaces

ADV_IMP.2 Implementation of the TSF

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_CCA.2.1D The developer shall conduct a search for covert channels for each information flow control policy.

AVA_CCA.2.2D The developer shall provide covert channel analysis documentation.

Content and presentation of evidence elements:

AVA_CCA.2.1C The analysis documentation shall identify covert channels and estimate their capacity.

AVA_CCA.2.2C The analysis documentation shall describe the procedures used for determining the existence of covert channels, and the information needed to carry out the covert channel analysis.

AVA_CCA.2.3C The analysis documentation shall describe all assumptions made during the covert channel analysis.

AVA_CCA.2.4C The analysis documentation shall describe the method used for estimating channel capacity, based on worst case scenarios.

AVA_CCA.2.5C The analysis documentation shall describe the worst case exploitation scenario for each identified covert channel.

AVA_CCA.2.6C **The analysis documentation shall provide evidence that the method used to identify covert channels is systematic.**

Evaluator action elements:

AVA_CCA.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_CCA.2.2E The evaluator shall confirm that the results of the covert channels analysis show that the TOE meets its functional requirements.

AVA_CCA.2.3E The evaluator shall selectively validate the covert channel analysis through testing.

AVA_CCA.3 Exhaustive covert channel analysis

Objectives

The objective is to identify covert channels that are identifiable through analysis.

In this component, the objective is to perform an exhaustive search for covert channels.

Application notes

Performing a covert channel analysis in an exhaustive way requires that additional evidence be provided that the plan that was followed for identifying covert channels is sufficient to ensure that all possible ways for covert channel exploration have been exercised.

Dependencies:

ADV_FSP.2 Fully defined external interfaces

ADV_IMP.2 Implementation of the TSF

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_CCA.3.1D The developer shall conduct a search for covert channels for each information flow control policy.

AVA_CCA.3.2D The developer shall provide covert channel analysis documentation.

Content and presentation of evidence elements:

AVA_CCA.3.1C The analysis documentation shall identify covert channels and estimate their capacity.

AVA_CCA.3.2C The analysis documentation shall describe the procedures used for determining the existence of covert channels, and the information needed to carry out the covert channel analysis.

AVA_CCA.3.3C The analysis documentation shall describe all assumptions made during the covert channel analysis.

AVA_CCA.3.4C The analysis documentation shall describe the method used for estimating channel capacity, based on worst case scenarios.

AVA_CCA.3.5C The analysis documentation shall describe the worst case exploitation scenario for each identified covert channel.

AVA_CCA.3.6C The analysis documentation shall provide evidence that the method used to identify covert channels is **exhaustive**.

Evaluator action elements:

AVA_CCA.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_CCA.3.2E The evaluator shall confirm that the results of the covert channels analysis show that the TOE meets its functional requirements.

AVA_CCA.3.3E The evaluator shall selectively validate the covert channel analysis through testing.

14.2 Misuse (AVA_MSU)

Objectives

Misuse investigates whether the TOE can be configured or used in a manner that is insecure but that an administrator or user of the TOE would reasonably believe to be secure.

The objectives are:

a) to minimise the probability of configuring or installing the TOE in a way that is insecure, without the end user or administrator being able to detect it;

b) to minimise the risk of human or other errors in operation that may deactivate, disable, or fail to activate security functions, resulting in an undetected insecure state.

Component levelling

The components are levelled on the increasing evidence to be provided by the developer and the increasing rigour of analysis.

Application notes

Conflicting, misleading, incomplete or unreasonable guidance may result in a user of the TOE believing that the TOE is secure when it is not, and can result in vulnerabilities.

An example of conflicting guidance would be two guidance instructions that imply different outcomes when the same input is supplied.

An example of misleading guidance would be the description of a single guidance instruction that could be parsed in more than one way, one of which may result in an insecure state.

An example of incomplete guidance would be a list of significant physical security requirements that omitted an important item, resulting in this item being overlooked by the administrator who believed the list to be complete.

An example of unreasonable guidance would be a recommendation to follow a procedure that imposed an unduly onerous administrative burden.

Guidance documentation is required. This may be contained in existing User or Administration documentation, or may be provided separately. If provided separately, the evaluators should confirm that the documentation is supplied with the TOE.

AVA_MSU.1 Examination of guidance

Objectives

The objective is to ensure that misleading, unreasonable and conflicting guidance is absent from the guidance documentation, and that secure procedures for all modes of operation have been addressed. Insecure states should be easy to detect.

Dependencies:

ADO_IGS.1 Installation, generation, and start-up procedures

ADV_FSP.1 Informal functional specification

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_MSU.1.1D **The developer shall provide guidance documentation.**

Content and presentation of evidence elements:

AVA_MSU.1.1C **The guidance documentation shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.**

AVA_MSU.1.2C **The guidance documentation shall be complete, clear, consistent and reasonable.**

AVA_MSU.1.3C **The guidance documentation shall list all assumptions about the intended environment.**

AVA_MSU.1.4C **The guidance documentation shall list all requirements for external security measures (including external procedural, physical and personnel controls).**

Evaluator action elements:

AVA_MSU.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

AVA_MSU.1.2E **The evaluator shall repeat all configuration and installation procedures to confirm that the TOE can be configured and used securely using only the supplied guidance documentation.**

AVA_MSU.1.3E **The evaluator shall determine that the use of the guidance documentation allows all insecure states to be detected.**

AVA_MSU.2 Validation of analysis

Objectives

The objective is to ensure that misleading, unreasonable and conflicting guidance is absent from the guidance documentation, and that secure procedures for all modes of operation have been addressed. Insecure states should be easy to detect. In this component, an analysis of the guidance documentation by the developer is required to provide additional assurance that the objective has been met.

Dependencies:

ADO_IGS.1 Installation, generation, and start-up procedures

ADV_FSP.1 Informal functional specification

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_MSU.2.1D The developer shall provide guidance documentation**.**

AVA_MSU.2.2D **The developer shall document an analysis of the guidance documentation.**

Content and presentation of evidence elements:

AVA_MSU.2.1C The guidance documentation shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

AVA_MSU.2.2C The guidance documentation shall be complete, clear, consistent and reasonable.

AVA_MSU.2.3C The guidance documentation shall list all assumptions about the intended environment.

AVA_MSU.2.4C The guidance documentation shall list all requirements for external security measures (including external procedural, physical and personnel controls).

AVA_MSU.2.5C **The developer's analysis documentation shall demonstrate that the guidance documentation is complete.**

Evaluator action elements:

AVA_MSU.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_MSU.2.2E The evaluator shall repeat all configuration and installation procedures, **and other procedures selectively**, to confirm that the TOE can be configured and used securely using only the supplied guidance documentation.

AVA_MSU.2.3E The evaluator shall determine that the use of the guidance documentation allows all insecure states to be detected.

AVA_MSU.2.4E **The evaluator shall confirm that the analysis shows that guidance is provided for secure operation in all modes of operation of the TOE.**

AVA_MSU.3 Analysis and testing for insecure states

Objectives

The objective is to ensure that misleading, unreasonable and conflicting guidance is absent from the guidance documentation, and that secure procedures for all modes of operation have been addressed. Insecure states should be easy to detect. In this component, an analysis of the guidance documentation by the developer is required to provide additional assurance that the objective has been met, and this analysis is validated and confirmed through testing by the evaluators.

Application notes

In this component the evaluator is required to undertake testing to ensure that if and when the TOE enters an insecure state this may easily be detected. This testing may be considered as a specific aspect of penetration testing.

Dependencies:

ADO_IGS.1 Installation, generation, and start-up procedures

ADV_FSP.1 Informal functional specification

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_MSU.3.1D The developer shall provide guidance documentation**.**

AVA_MSU.3.2D The developer shall document an analysis of the guidance documentation.

Content and presentation of evidence elements:

AVA_MSU.3.1C The guidance documentation shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

AVA_MSU.3.2C The guidance documentation shall be complete, clear, consistent and reasonable.

AVA_MSU.3.3C The guidance documentation shall list all assumptions about the intended environment.

AVA_MSU.3.4C The guidance documentation shall list all requirements for external security measures (including external procedural, physical and personnel controls).

AVA_MSU.3.5C The developer's analysis documentation shall demonstrate that the guidance documentation is complete.

Evaluator action elements:

AVA_MSU.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_MSU.3.2E The evaluator shall repeat all configuration and installation procedures, and other procedures selectively, to confirm that the TOE can be configured and used securely using only the supplied guidance documentation.

AVA_MSU.3.3E The evaluator shall determine that the use of the guidance documentation allows all insecure states to be detected.

AVA_MSU.3.4E The evaluator shall confirm that the analysis shows that guidance is provided for secure operation in all modes of operation of the TOE.

AVA_MSU.3.5E **The evaluator shall perform independent testing to confirm that an administrator or user, with an understanding of the guidance documentation, would reasonably be able to determine if the TOE is configured and operating in a manner that is insecure.**

14.3 Strength of TOE security functions (AVA_SOF)

Objectives

Even if a TOE security function cannot be bypassed, deactivated, or corrupted, it may still be possible to defeat it because there is a vulnerability in the concept of its underlying security mechanisms. For those functions a qualification of their security behaviour can be made using the results of a quantitative or statistical analysis of the security behaviour of these mechanisms and the effort required to overcome them. The qualification is made in the form of a strength of TOE security functions claim.

Component levelling

There is only one component in this family.

Application notes

Security functions are implemented by security mechanisms. For example, a password mechanism can be used in the implementation of the identification and authentication security function.

The strength of TOE security functions evaluation is performed at the level of the security mechanism, but its results provide knowledge about the ability of the related security function to counter the identified threats.

The strength of TOE security function analysis should consider at least the contents of all the TOE deliverables, including the ST, for the targeted evaluation assurance level.

AVA_SOF.1 Strength of TOE security function evaluation

Dependencies:

ADV_FSP.1 Informal functional specification

ADV_HLD.1 Descriptive high-level design

Developer action elements:

AVA_SOF.1.1D **The developer shall perform a strength of TOE security function analysis for each mechanism identified in the ST as having a strength of TOE security function claim.**

Content and presentation of evidence elements:

AVA_SOF.1.1C **For each mechanism with a strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the minimum strength level defined in the PP/ST.**

AVA_SOF.1.2C **For each mechanism with a specific strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the specific strength of function metric defined in the PP/ST.**

Evaluator action elements:

AVA_SOF.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

AVA_SOF.1.2E **The evaluator shall confirm that the strength claims are correct.**

14.4 Vulnerability analysis (AVA_VLA)

Objectives

Vulnerability analysis is an assessment to determine whether vulnerabilities identified, during the evaluation of the construction and anticipated operation of the TOE or by other methods (e.g. by flaw hypotheses), could allow users to violate the TSP.

Vulnerability analysis deals with the threats that a user will be able to discover flaws that will allow unauthorised access to resources (e.g. data), allow the ability to interfere with or alter the TSF, or interfere with the authorised capabilities of other users.

Component levelling

Levelling is based on an increasing rigour of vulnerability analysis by the developer and the evaluator.

Application notes

The developer is required to document the disposition of identified vulnerabilities to allow the evaluator to make use of that information if it is found useful as a support for the evaluator's independent vulnerability analysis.

The vulnerability analysis should consider at least the contents of all the TOE deliverables including the ST for the targeted evaluation assurance level.

Obvious vulnerabilities are those that allow common attacks or those that might be suggested by the TOE interface description. Obvious vulnerabilities include those in the public domain, details of which should be known to a developer or available from an evaluation authority.

Obvious penetration attacks are those that are open to exploitation that requires a minimum of understanding of the TOE, skill, technical sophistication, and resources.

Independent vulnerability analysis is based on highly detailed technical information. The attacker is assumed to be thoroughly familiar with the specific implementation of the TOE. The attacker is presumed to have a high level of technical sophistication.

Performing a search for vulnerabilities in a systematic way requires that the developer identify those vulnerabilities in a structured and repeatable way, as opposed to identifying them in an ad-hoc fashion.

The evidence identifies all the TOE documentation upon which the search for flaws was based.

A vulnerability analysis is performed by the developer to ascertain the presence of security vulnerabilities.

The intent of the developer analysis is to confirm that no identified security vulnerabilities can be exploited in the intended environment for the TOE and that the TOE is resistant to obvious penetration attacks.

An independent vulnerability analysis is performed by the evaluator, which goes beyond the vulnerabilities identified by the developer.

The main intent of the evaluator analysis is to determine that the TOE is resistant to penetration attacks performed by an attacker possessing a low (for AVA_VLA.2), moderate (for AVA_VLA.3) or high (for AVA_VLA.4) attack potential. To accomplish this intent, the evaluator first assesses the exploitability of all identified vulnerabilities. This is accomplished by conducting penetration testing. The evaluator should assume the role of an attacker with a low (for AVA_VLA.2), moderate (for AVA_VLA.3) or high (for AVA_VLA.4) attack potential when attempting to penetrate the TOE. Any exploitation of vulnerabilities by such an attacker should be considered “obvious penetration attacks” in the context of this component.

AVA_VLA.1 Developer vulnerability analysis

Objectives

A vulnerability analysis is performed by the developer to ascertain the presence of obvious security vulnerabilities, and to confirm that they cannot be exploited in the intended environment for the TOE.

Application notes

The evaluator should consider performing additional tests as a result of potential exploitable vulnerabilities identified during other parts of the evaluation.

Dependencies:

ADV_FSP.1 Informal functional specification

ADV_HLD.1 Descriptive high-level design

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_VLA.1.1D **The developer shall perform and document an analysis of the TOE deliverables searching for obvious ways in which a user can violate the TSP.**

AVA_VLA.1.2D **The developer shall document the disposition of obvious vulnerabilities.**

Content and presentation of evidence elements:

AVA_VLA.1.1C **The documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.**

Evaluator action elements:

AVA_VLA.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

AVA_VLA.1.2E **The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure obvious vulnerabilities have been addressed.**

AVA_VLA.2 Independent vulnerability analysis

Objectives

A vulnerability analysis is performed by the developer to ascertain the presence of security vulnerabilities, and to confirm that they cannot be exploited in the intended environment for the TOE.

The evaluator performs independent penetration testing, supported by the evaluator's independent vulnerability analysis, to determine that the TOE is resistant to penetration attacks performed by attackers possessing a low attack potential.

Dependencies:

ADV_FSP.1 Informal functional specification

ADV_HLD.2 Security enforcing high-level design

ADV_IMP.1 Subset of the implementation of the TSF

ADV_LLD.1 Descriptive low-level design

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_VLA.2.1D The developer shall perform and document an analysis of the TOE deliverables searching for **ways** in which a user can violate the TSP.

AVA_VLA.2.2D The developer shall document the disposition of **identified** vulnerabilities.

Content and presentation of evidence elements:

AVA_VLA.2.1C The documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.

AVA_VLA.2.2C **The documentation shall justify that the TOE, with the identified vulnerabilities, is resistant to obvious penetration attacks**.

Evaluator action elements:

AVA_VLA.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.2.2E The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure **the identified** vulnerabilities have been addressed.

AVA_VLA.2.3E **The evaluator shall perform an independent vulnerability analysis.**

AVA_VLA.2.4E **The evaluator shall perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of additional identified vulnerabilities in the intended environment.**

AVA_VLA.2.5E **The evaluator shall determine that the TOE is resistant to penetration attacks performed by an attacker possessing a low attack potential.**

AVA_VLA.3 Moderately resistant

Objectives

A vulnerability analysis is performed by the developer to ascertain the presence of security vulnerabilities, and to confirm that they cannot be exploited in the intended environment for the TOE.

The evaluator performs independent penetration testing, supported by the evaluator’s independent vulnerability analysis, to determine that the TOE is resistant to penetration attacks performed by attackers possessing a moderate attack potential.

ADV_FSP.1 Informal functional specification

ADV_HLD.2 Security enforcing high-level design

ADV_IMP.1 Subset of the implementation of the TSF

ADV_LLD.1 Descriptive low-level design

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Content and presentation of evidence elements:

AVA_VLA.3.1D The developer shall perform and document an analysis of the TOE deliverables searching for ways in which a user can violate the TSP.

AVA_VLA.3.2D The developer shall document the disposition of identified vulnerabilities.

Content and presentation of evidence elements:

AVA_VLA.3.1C The documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.

AVA_VLA.3.2C The documentation shall justify that the TOE, with the identified vulnerabilities, is resistant to obvious penetration attacks.

AVA_VLA.3.3C **The evidence shall show that the search for vulnerabilities is systematic.**

Evaluator action elements:

AVA_VLA.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.3.2E The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure the identified vulnerabilities have been addressed.

AVA_VLA.3.3E The evaluator shall perform an independent vulnerability analysis.

AVA_VLA.3.4E The evaluator shall perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of additional identified vulnerabilities in the intended environment.

AVA_VLA.3.5E The evaluator shall determine that the TOE is resistant to penetration attacks performed by an attacker possessing a **moderate** attack potential.

AVA_VLA.4 Highly resistant

Objectives

A vulnerability analysis is performed by the developer to ascertain the presence of security vulnerabilities, and to confirm that they cannot be exploited in the intended environment for the TOE.

The evaluator performs independent penetration testing, supported by the evaluator's independent vulnerability analysis, to determine that the TOE is resistant to penetration attacks performed by attackers possessing a high attack potential.

Dependencies:

ADV_FSP.1 Informal functional specification

ADV_HLD.2 Security enforcing high-level design

ADV_IMP.1 Subset of the implementation of the TSF

ADV_LLD.1 Descriptive low-level design

AGD_ADM.1 Administrator guidance

AGD_USR.1 User guidance

Developer action elements:

AVA_VLA.4.1D The developer shall perform and document an analysis of the TOE deliverables searching for ways in which a user can violate the TSP.

AVA_VLA.4.2D The developer shall document the disposition of identified vulnerabilities.

Content and presentation of evidence elements:

AVA_VLA.4.1C The documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.

AVA_VLA.4.2C The documentation shall justify that the TOE, with the identified vulnerabilities, is resistant to obvious penetration attacks.

AVA_VLA.4.3C The evidence shall show that the search for vulnerabilities is systematic.

AVA_VLA.4.4C **The analysis documentation shall provide a justification that the analysis completely addresses the TOE deliverables.**

Evaluator action elements:

AVA_VLA.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.4.2E The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure the identified vulnerabilities have been addressed.

AVA_VLA.4.3E The evaluator shall perform an independent vulnerability analysis.

AVA_VLA.4.4E The evaluator shall perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of additional identified vulnerabilities in the intended environment.

AVA_VLA.4.5E The evaluator shall determine that the TOE is resistant to penetration attacks performed by an attacker possessing a **high** attack potential.

15 Assurance maintenance paradigm

15.1 Introduction

This chapter provides the discourse on an assurance maintenance paradigm that is supported by the Maintenance of assurance class (AMA). As such it provides helpful information to understand one possible approach to applying the AMA requirements.

Maintenance of assurance is a concept intended to be applied after a TOE has been evaluated and certified against the criteria in chapters 4-5 and 8-14. The maintenance of assurance requirements are aimed at assuring that the TOE will continue to meet its security target as changes are made to the TOE or its environment. Such changes include the discovery of new threats or vulnerabilities, changes in user requirements, the correction of bugs found in the certified TOE, and other updates to the functionality provided.

One way to determine that assurance has been maintained is by a re-evaluation of the TOE. The term 're-evaluation' here refers to an evaluation of a new version of the TOE that addresses all security relevant changes made to the certified version of the TOE and re-uses previous evaluation results where these are still valid. However, in many cases it is unlikely to be practical to perform a re-evaluation of every new version of the TOE in order to ensure that assurance continues to be maintained.

The main goal of class AMA is therefore to define a set of requirements which can be applied to provide confidence that the assurance established in a TOE is being maintained, without always requiring a formal re-evaluation of new versions of the TOE. Class AMA does not remove entirely the need for re-evaluation. In some cases, changes may be so significant that only a re-evaluation can be relied upon to ensure that assurance has been maintained. The requirements of this class thus have a secondary goal of supporting cost-effective re-evaluation of a TOE when this is necessary.

It should be noted that it is possible to re-evaluate any new version of a TOE against the criteria in chapters 4-5 and 8-14 without any of the AMA requirements having been satisfied. However, class AMA includes requirements which can be used in support of any such re-evaluation.

Maintenance developer and evaluator actions are intended to be applied after the TOE has been evaluated and certified although, as described below, some requirements can be applied at the time of the evaluation. For clarity, the following terms are used in this paradigm description:

a) the *certified version* of the TOE refers to the version that has been evaluated and certified;

b) the *current version* of the TOE refers to a version that differs in some respect from the certified version; this could be, for example:

- a new release of the TOE
- the certified version with patches applied to correct subsequently discovered bugs
- the same basic version of the TOE, but on a different hardware or software platform.

The developer and evaluator roles in this class are as described in Part 1 of the criteria. However, it is not necessarily the case that the evaluator referred to in the requirements of this class will be the same as that which evaluated the certified version of the TOE.

In order to allow assurance to be maintained in a TOE without always requiring a formal re-evaluation, the requirements in this class place an obligation on the developer to maintain evidence that shows that the TOE continues to satisfy its security target (e.g. evidence of developer testing).

15.2 Assurance maintenance cycle

This section describes one possible approach to the use of the assurance maintenance families and components, intended to illustrate use of the concepts. The example is modeled on an ‘assurance maintenance cycle’ that may be divided into the following three phases:

a) the *acceptance phase*, at the start of a cycle, in which the developer’s plans and procedures for assurance maintenance during the cycle are established by the developer and independently validated by an evaluator;

b) the *monitoring phase*, in which the developer provides at one or more points during the cycle evidence that the assurance in the TOE is being maintained in accordance with the established plans and procedures, this evidence of assurance maintenance being independently checked by an evaluator;

c) the *re-evaluation phase*, completing the cycle, in which an updated version of the TOE is submitted for a re-evaluation based on the changes affecting the TOE since the certified version.

The families within AMA address primarily the first two of these phases, while providing support for the third. These phases are introduced here simply to help describe the application of the assurance maintenance requirements. There is no intention to mandate an assurance maintenance scheme which formally incorporates these phases.

The assurance maintenance cycle is illustrated in Figure 15.1 below.

In this example, a TOE can enter the monitoring phase only when the acceptance phase has been successfully concluded (i.e. the developer’s plans and procedures for assurance maintenance have been accepted). If the developer makes changes to these plans or procedures during the monitoring phase then the TOE will need to re-enter the acceptance phase to get the changes accepted.

During the monitoring phase the developer follows the assurance maintenance plans and procedures, conducting an analysis of the security impact of changes affecting the TOE (security impact analysis). At certain points during this phase, an evaluator independently checks (by means of an audit) the developer’s work. The developer is required to ensure that the plans and procedures are followed, and that security impact analysis is performed correctly.

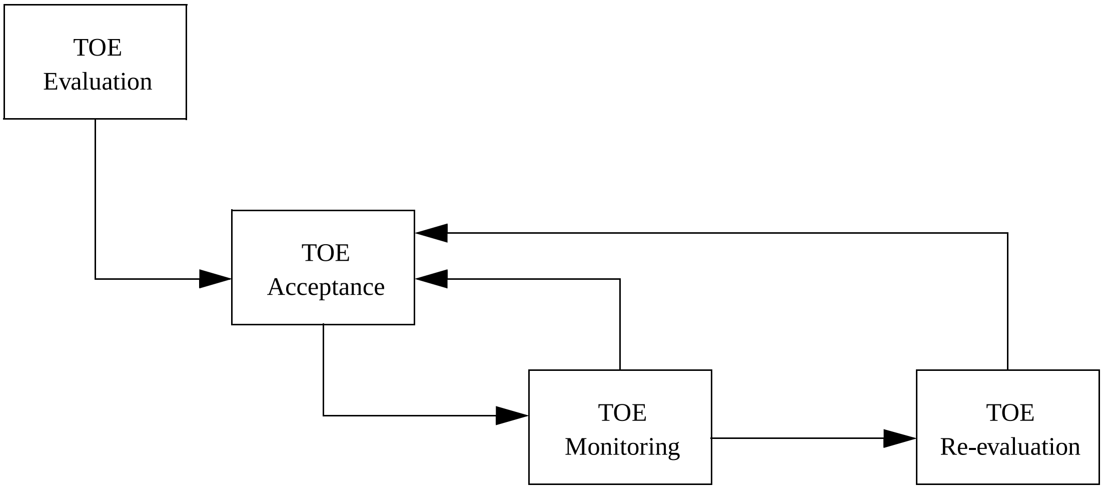

Figure 15.1 - Example assurance maintenance cycle

Therefore, once a TOE is in the monitoring phase, it becomes possible to have confidence that the assurance in the TOE has been maintained for new versions of the TOE produced by the developer.

A TOE that is subject to change would not continue in the monitoring phase for an indefinite period: at some point a re-evaluation of the TOE would be necessary. The decision as to when a re-evaluation would be required is dependent on cumulative changes to the TOE as well as especially significant changes. For example, a large number of minor changes could have an impact on assurance equivalent to that of a major change. The developer's assurance maintenance plan defines the scope of the changes that may be made to the TOE during the monitoring phase (see section 15.3.1 below).

In a similar way, it would not possible to 'uprate' a TOE (i.e. increase the assurance level) during the monitoring phase: this could only be achieved by means of an evaluation of the TOE (making appropriate reuse of previous evaluation results).

The assurance maintenance status of the TOE will have to be reviewed if it is discovered that the assurance maintenance procedures are not being followed, and that as a result assurance in the TOE is undermined. In some cases the developer may be required to submit the TOE for re-evaluation, and afterwards start a new assurance maintenance cycle.

15.2.1 TOE acceptance

In the example, the TOE acceptance phase of the assurance maintenance cycle can be refined into the following, which uses the assurance maintenance plan and TOE component categorisation report families from the AMA class.

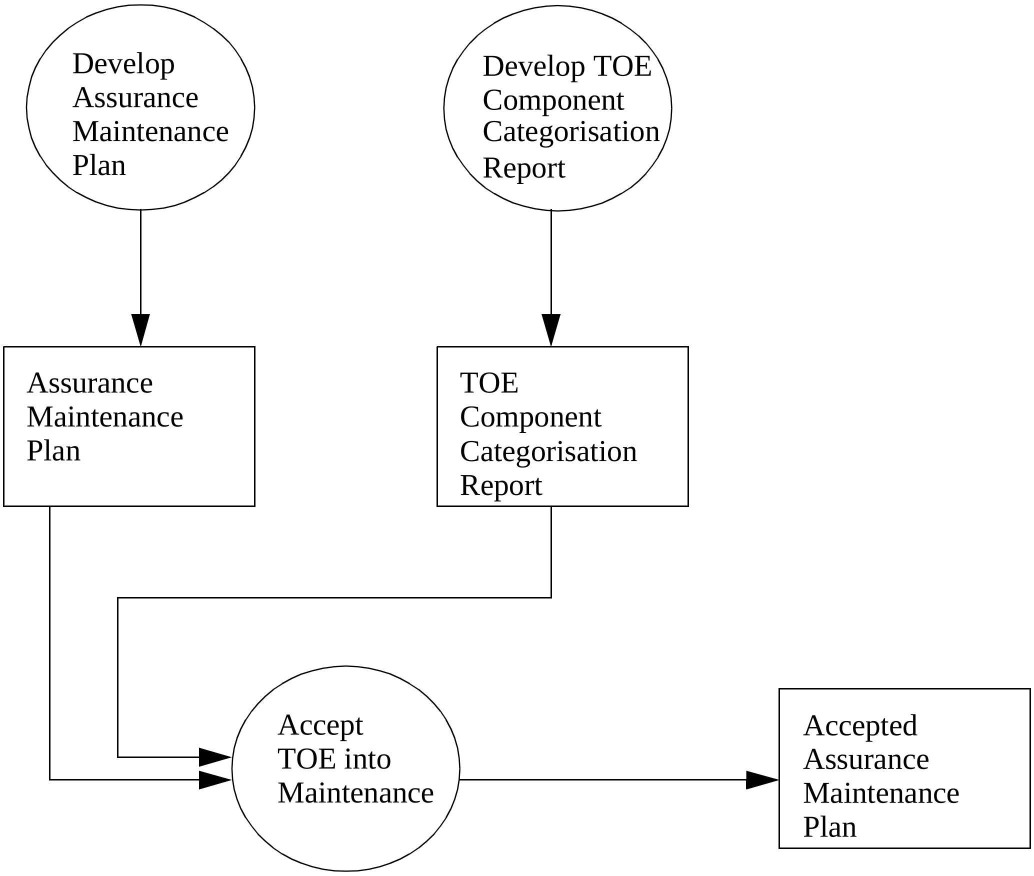
Develop Assurance Maintenance Plan
Develop TOE Component Categorisation Report
Assurance Maintenance Plan
TOE Component Categorisation Report
Accept TOE into Maintenance
Accepted Assurance Maintenance Plan

Figure 15.2 - Example TOE acceptance approach

15.2.2 TOE monitoring

The TOE monitoring phase of the assurance maintenance cycle would be refined into the following, which uses the Evidence of assurance maintenance and Security impact analysis families of the AMA Class.

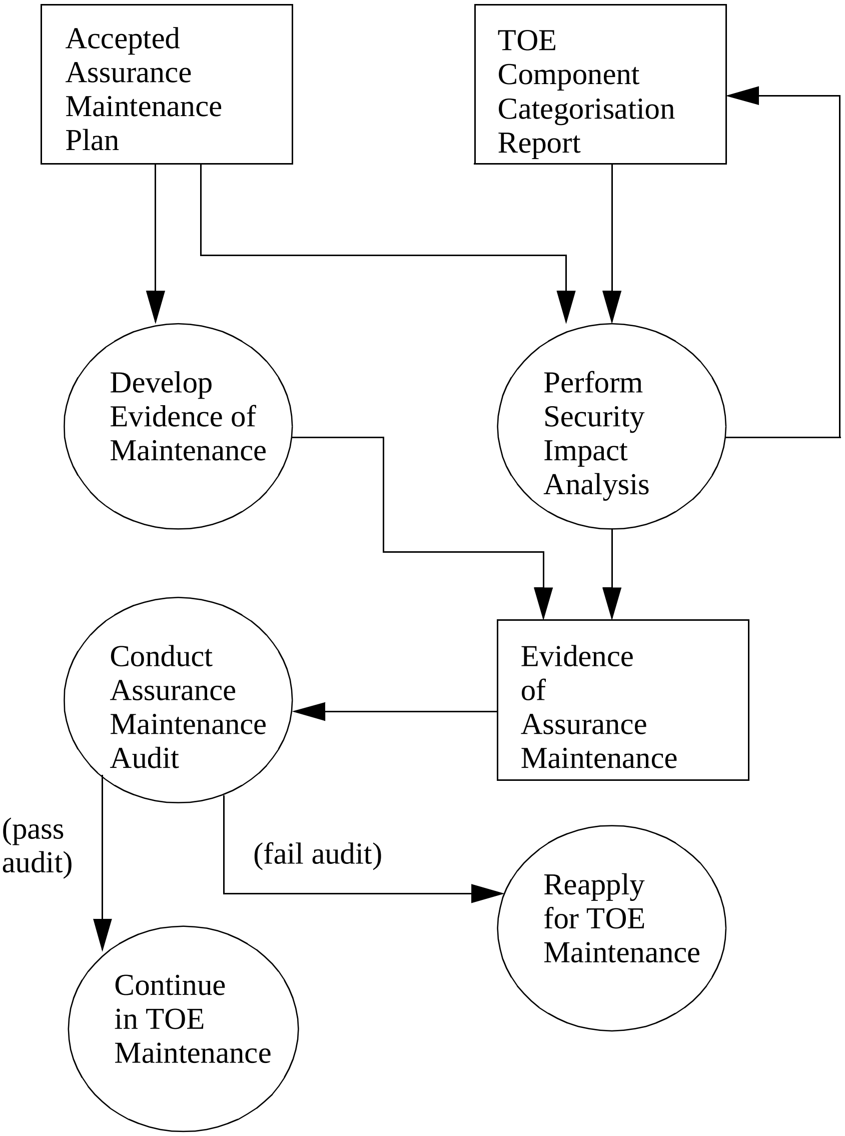
Accepted Assurance Maintenance Plan
TOE Component Categorisation Report
Develop Evidence of Maintenance
Perform Security Impact Analysis
Conduct Assurance Maintenance Audit
Evidence of Assurance Maintenance
(pass audit)
(fail audit)
Reapply for TOE Maintenance
Continue in TOE Maintenance

Figure 15.3 - Example TOE monitoring approach

15.2.3 Re-evaluation

The third phase of this example maintenance cycle is the re-evaluation phase, in which the evaluator makes use of the impact analysis and evidence of assurance maintenance to re-examine parts of the TOE, using the assurance components applicable for the target assurance level.

Re-evaluation activities would be scheduled in the AM Plan, or could be required in response to unforseen significant changes to the TOE or its environment for which assurance maintenance activities were considered inappropriate.

15.3 Assurance maintenance class and families

To support assurance maintenance approaches the class AMA has been developed, and comprises four families as shown in Table 15.1

Assurance Class	Assurance Family	Abbreviated Name
Class AMA: Maintenance of assurance	Assurance maintenance plan	AMA_AMP
	TOE component categorisation report	AMA_CAT
	Evidence of assurance maintenance	AMA_EVD
	Security impact analysis	AMA_SIA

Table 15.1 - Maintenance of assurance family breakdown and mapping

15.3.1 Assurance maintenance plan

The AM Plan provides a clear identification of the baseline for assurance maintenance, in terms of the evaluation results and the definition of the categorisation of TOE components.

The Assurance Maintenance Plan (AM Plan) identifies the plans and procedures a developer implements in order to ensure that the assurance that was established in the certified TOE is maintained as changes are made to the TOE or its environment. An AM Plan covers one assurance maintenance cycle.

The AM Plan defines the scope of changes that can be made to the TOE without triggering a re-evaluation. The specific approach to be followed is scheme dependent, but the following types of change are likely to be outside the scope of the AM Plan and thus might only be addressed by means of a re-evaluation:

a) significant changes to the security target (i.e. significant changes to the security environment, security objectives or security functional requirements, or *any* increase in the assurance requirements);

b) significant changes to external TSF interfaces categorised as TSP-enforcing;

c) (where the assurance requirements include ADV_HLD.1 or higher components) significant changes to TSF subsystems categorised as TSP-enforcing.

A more precise specification of the rules is outside the scope of the CC, not least because the definition of what constitutes a *significant* change will be dependent on the type of TOE evaluated, and on the content of the security target.

The AM Plan is required to define or reference the procedures that will be applied to ensure that assurance in the TOE is maintained during the assurance maintenance cycle. Four types of procedure are identified that should be applied:

a) configuration management procedures, controlling and recording changes to the TOE in support of the developer’s security impact analysis, as well as supporting documentation (including the AM Plan itself);

b) procedures to maintain ‘assurance evidence’ (i.e. the maintenance of documentary evidence as required by the appropriate assurance requirements), a key aspect of which is functional testing of the security functions of the TOE, and the developer’s regression testing policy in particular;

c) procedures governing the security impact analysis of changes affecting the TOE (Note that this includes changes within the TOE environment, such as new threats or attack methods that may need to be identified and tracked), and the maintenance of the TOE component categorisation report as changes are made;

d) flaw remediation procedures, covering the tracking and correction of reported security flaws (as required by ALC_FLR.1).

The AM Plan is expected to remain valid until completion of the assurance maintenance cycle (i.e. completion of the scheduled re-evaluation), after which a new AM Plan will be required. The AM Plan is expected to be invalidated if the developer does not follow the plan, or makes changes to the TOE that are outside the scope of the plan, or has to make such changes in order for the TOE to remain effective within its environment. An updated AM Plan should be re-submitted and accepted before a TOE enters a new monitoring phase.

The AM Plan requires the developer to identify a developer security analyst whose responsibility is to monitor the assurance maintenance process. The role may be filled by more than one individual. The developer security analyst is required to be familiar with the TOE, the evaluation results and applicable assurance requirements as an essential prerequisite for fulfilling the role. The requirements do not specify how this level of knowledge and experience should be gained; however, it is likely that a prospective developer security analyst will have to undergo some form of training programme to address any deficiencies in his or her knowledge and experience. The developer security analyst needs to have sufficient authority within the developer’s organisation to ensure that the requirements of the AM Plan and its associated procedures are followed.

15.3.2 TOE component categorisation report

The aim of the TOE component categorisation report is to complement the AM Plan by providing a categorisation of the components of a TOE (e.g. TSF subsystems) according to their relevance to security. This categorisation acts as a focus for the developer’s security impact analysis, and also for the subsequent re-evaluation of the TOE.

The checking of the TOE component categorisation report occurs during the acceptance phase; the evaluator checks are applied only in respect of the version of the report for the certified version of the TOE. While the assurance maintenance procedures identified in the AM Plan require the developer to update the TOE component categorisation report as changes are made to the TOE, the evaluators are not required to re-review the document; however, any such updates are likely to be inspected during the monitoring phase.

The TOE component categorisation report covers all TSF representations for the level of assurance being maintained. The TOE component categorisation report also identifies:

a) any hardware, firmware or software components that are external to the TOE (e.g. hardware or software platforms), and that satisfy IT security requirements as defined in the ST;

b) any development tools that, if modified, will have an impact on the required assurance that the TOE satisfies its ST.

The TOE component categorisation report also provides a description of the approach used for the categorisation of TOE components. As a minimum, TOE components are required to be categorised as either TSP-enforcing or non-TSP-enforcing. The description of the categorisation scheme is intended to enable the developer security analyst to decide the category to which any new TOE component should be assigned, and also when to change the category of an existing TOE component following changes to the TOE or its ST.

The initial categorisation of the components of the TOE will be based on evidence provided by the developer in support of the evaluation of the TOE, independently validated by the evaluators. Although maintenance of the document is the responsibility of the developer security analyst, its initial contents may be based on the results of the evaluation of the TOE.

It may be useful for the ST to include AMA_CAT.1 where there is a requirement that assurance be maintained in future versions of the TOE. This applies irrespective of whether assurance maintenance is to be achieved by application of the requirements in this class, or by periodic re-evaluations of the TOE.

15.3.3 Evidence of assurance maintenance

Confidence needs to be established that the assurance in the TOE is being maintained by the developer, in accordance with the AM Plan. This is achieved through the provision of evidence that demonstrates that the assurance in the TOE has been maintained, which is independently checked by an evaluator. This check (termed an ‘AM audit’) would typically be applied periodically during the monitoring phase of the TOE’s assurance maintenance cycle.

AM audits are conducted in accordance with the schedule defined in the AM Plan. The developer and evaluator actions required by AMA_EVD.1 will therefore be invoked one or more times during the monitoring phase of the assurance maintenance cycle. The evaluators may need to visit the TOE development environment to examine the required evidence, but other ways of performing the checks are not precluded.

The developer is required to provide evidence that the assurance maintenance procedures referred to in the AM Plan are being followed. This will include:

a) configuration management records;

b) documentation referenced by the security impact analysis, including the current version of the TOE component categorisation report, and evidence for all applicable assurance requirements such as design updates, test documentation, new versions of guidance documents, and so on;

c) evidence of the tracking of security flaws.

The evaluator’s check of the developer’s security impact analysis (required by AMA_SIA.1 on which AMA_EVD.1 depends) will act as a focus for the AM audit. The AM audit will, in turn, provide corroboration of the developer’s analysis (and hence confidence in the quality of the analysis), thereby serving to validate the developer’s claim that assurance has been maintained in the current version of the TOE.

An AM audit requires the evaluators to confirm that functional testing has been performed on the current version of the TOE. This is highlighted as a separate check because test documentation provides firm evidence that the TOE security functions continue to operate as specified. The evaluators sample the test documentation to confirm that the developer testing shows that the security functions operate as specified, and that the coverage and depth of testing is commensurate with the level of assurance being maintained.

15.3.4 Security impact analysis

The aim of the security impact analysis is to provide confidence that assurance has been maintained in the TOE, through an analysis performed by the developer of the security impact of all changes affecting the TOE since it was certified. These requirements may be applied during a monitoring phase or a re-evaluation phase.

The developer’s security impact analysis is based on the TOE component categorisation report: changes to TSP-enforcing TOE components may have an impact on the assurance that the TOE continues to meet its ST following the changes. All such changes therefore require an analysis of their security impact to show that they do not undermine assurance in the TOE.

The components in this family may be used in support of either a subsequent AM audit or a re-evaluation of the TOE.

For an AM audit, the evaluators’ review of the security impact analysis should act as a focus for the subsequent audit activities, which should in turn provide corroboration of the developer’s analysis.

The security impact analysis identifies the changes from the certified version of the TOE, in terms of the TOE components which are either new, or which have been modified. The evaluators check the accuracy of this information during either the associated AM audit, or the associated re-evaluation of the TOE.

Provision of the security impact analysis in support of a re-evaluation should reduce the level of evaluator effort needed to establish the required level of assurance in the TOE. Application of AMA_SIA.2, which requires a full examination of the security impact analysis, is likely to provide maximum benefit to the re-evaluation. The precise detailed conditions under which an evaluation authority might wish the security impact analysis to be used in practice in a re-evaluation are beyond the scope of the CC.

16 Class AMA: Maintenance of assurance

The maintenance of assurance class provides requirements that are intended to be applied after a TOE has been certified against the CC. These requirements are aimed at assuring that the TOE will continue to meet its security target as changes are made to the TOE or its environment. Such changes include the discovery of new threats or vulnerabilities, changes in user requirements, and the correction of bugs found in the certified TOE.

The class comprises four families, and the hierarchy of components within, as shown in Figure 16.1:

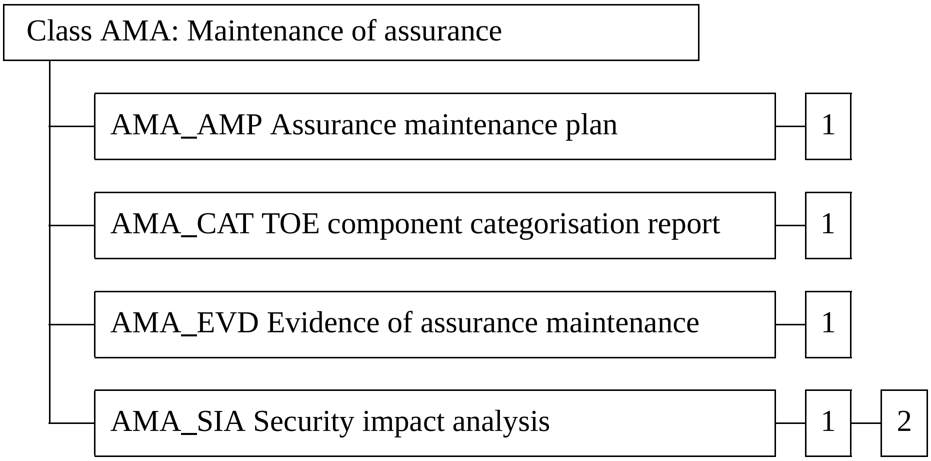
Class AMA: Maintenance of assurance
AMA_AMP Assurance maintenance plan
1
AMA_CAT TOE component categorisation report
1
AMA_EVD Evidence of assurance maintenance
1
AMA_SIA Security impact analysis
1
2

Figure 16.1 - Maintenance of assurance class decomposition

16.1 Assurance maintenance plan (AMA_AMP)

Objectives

The Assurance Maintenance Plan (AM Plan) identifies the plans and procedures a developer must implement in order to ensure that the assurance that was established in the certified TOE is maintained as changes are made to the TOE or its environment. The AM Plan is specific to the TOE, and is tailored to the developer's own practices and procedures.

Component levelling

This family contains only one component.

Application notes

An AM Plan covers one assurance maintenance cycle, this being the period from the completion of the most recent evaluation of the TOE to the completion of the next planned re-evaluation.

The requirements AMA_AMP.1.2C and AMA_AMP.1.3C serve to provide a clear identification of the baseline for assurance maintenance, in terms of the evaluation results and the definition of the categorisation of TOE components. The TOE component categorisation report is subject to the requirements of the AMA_CAT family, and provides the basis for the security impact analysis performed by the developer security analyst.

The definition of the scope of changes covered by the plan, as required by AMA_AMP.1.4C, should be in terms of the category of components of the TOE that may be changed and the representational level at which changes can occur (referencing the TOE component categorisation report where appropriate).

AMA_AMP.1.5C requires a description of the developer's *current* plans for any new releases of the TOE. These plans may be subject to change, and hence require an update to the AM Plan. It should be noted, however, that in this context the term *new release* does not, for example, include minor ('unplanned') releases of the TOE to incorporate bug fixes.

AMA_AMP.1.6C requires a definition of the planned schedule for AM audits (see the AMA_EVD family below) and the targeted re-evaluation of the TOE, together with a justification of the proposed schedules. The schedules may be defined in terms of elapsed time (e.g. annual AM audits), or they may be linked to specific new releases of the TOE. The planned schedules should take into account the expected changes to the TOE during the period, and also any elapsed period between the evaluation of the TOE and the establishment of the AM Plan. In particular, any changes outside the scope of the AM Plan will trigger a re-evaluation.

AMA_AMP.1 Assurance maintenance plan

Dependencies:

ACM_CAP.2 Configuration items

ALC_FLR.1 Basic flaw remediation

AMA_CAT.1 TOE component categorisation report

Developer action elements:

AMA_AMP.1.1D **The developer shall provide an AM Plan**.

Content and presentation of evidence elements:

AMA_AMP.1.1C **The AM Plan shall contain or reference a brief description of the TOE, including the security functionality it provides.**

AMA_AMP.1.2C **The AM Plan shall identify the certified version of the TOE, and shall reference the evaluation results.**

AMA_AMP.1.3C **The AM Plan shall reference the TOE component categorisation report for the certified version of the TOE.**

AMA_AMP.1.4C **The AM Plan shall define the scope of changes to the TOE that are covered by the plan.**

AMA_AMP.1.5C **The AM Plan shall describe the TOE life-cycle, and shall identify the current plans for any new releases of the TOE, together with a brief description of any planned changes that are likely to have a significant security impact.**

AMA_AMP.1.6C **The AM Plan shall describe the assurance maintenance cycle, stating and justifying the planned schedule of AM audits and the target date of the next re-evaluation of the TOE.**

AMA_AMP.1.7C **The AM Plan shall identify the individual(s) who will assume the role of developer security analyst for the TOE.**

AMA_AMP.1.8C **The AM Plan shall describe how the developer security analyst role will ensure that the procedures documented or referenced in the AM Plan are followed.**

AMA_AMP.1.9C **The AM Plan shall describe how the developer security analyst role will ensure that all developer actions involved in the analysis of the security impact of changes affecting the TOE are performed correctly.**

AMA_AMP.1.10C **The AM Plan shall justify why the identified developer security analyst(s) have sufficient familiarity with the security target, functional specification and (where appropriate) high-level design of the TOE, and with the evaluation results and all applicable assurance requirements for the certified version of the TOE.**

AMA_AMP.1.11C **The AM Plan shall describe or reference the procedures to be applied to maintain the assurance in the TOE, which as a minimum shall include the procedures for configuration management, maintenance of assurance evidence, performance of the analysis of the security impact of changes affecting the TOE, and flaw remediation.**

Evaluator action elements:

AMA_AMP.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

AMA_AMP.1.2E **The evaluator shall confirm that the proposed schedules for AM audits and re-evaluation of the TOE are acceptable and consistent with the proposed changes to the TOE.**

16.2 TOE component categorisation report (AMA_CAT)

Objectives

The aim of the TOE component categorisation report is to complement the AM Plan by providing a categorisation of the components of a TOE (e.g. TSF subsystems) according to their relevance to security. This categorisation acts as a focus for the developer's security impact analysis, and also for the subsequent re-evaluation of the TOE.

Component levelling

This family contains only one component.

Application notes

The term "least abstract TSF representation" in AMA_CAT.1.1 refers to the least abstract representation of the TSF that was provided for the level of assurance that is being maintained. For example, if the TOE is to be maintained at an assurance level of EAL3, then the least abstract TSF representation is the high-level design, and the following TOE components must be categorised:

a) all external TSF interfaces identifiable in the functional specification;

b) all TSF subsystems identifiable in the high-level design.

While AMA_CAT requires at least two categories to be defined, it may be appropriate (dependent on the type of TOE) to further subdivide the TSP-enforcing category in order to help focus the developer's security impact analysis. For example, TSP-enforcing components could be categorised as either *security critical* or *security supporting* where:

a) security critical TOE components are those which are *directly* responsible for the enforcement of at least one IT security function defined in the security target;

b) security supporting TOE components are those which are not *directly* responsible for the enforcement of any IT security function (and hence are not security critical), but which are nonetheless relied upon to uphold the IT security functions; this category may in turn include two distinct types of TOE component:

  - those that provide services to security critical TOE components, and hence are relied upon to function correctly;
  - those that do not provide any such service, but which nonetheless have to be trusted not to behave in a malicious manner (i.e. introducing a vulnerability).

AMA_CAT.1.3C requires an identification of any development tools that, if modified, will have an impact on the assurance that the TOE satisfies its security target (e.g. the compiler used to create the object code).

AMA_CAT.1 TOE component categorisation report

Dependencies:

ACM_CAP.2 Configuration items

Developer action elements:

AMA_CAT.1.1D **The developer shall provide a TOE component categorisation report for the certified version of the TOE.**

Content and presentation of evidence elements:

AMA_CAT.1.1C **The TOE component categorisation report shall categorise each component of the TOE, identifiable in each TSF representation from the most abstract to the least abstract, according to its relevance to security; as a minimum, TOE components must be categorised as one of TSP-enforcing or non-TSP-enforcing.**

AMA_CAT.1.2C **The TOE component categorisation report shall describe the categorisation scheme used, so that it can be determined how to categorise new components introduced into the TOE, and also when to re-categorise existing TOE components following changes to the TOE or its security target.**

AMA_CAT.1.3C **The TOE component categorisation report shall identify any tools used in the development environment that, if modified, will have an impact on the assurance that the TOE satisfies its security target.**

Evaluator action elements:

AMA_CAT.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

AMA_CAT.1.2E **The evaluator shall confirm that the categorisation of TOE components and tools, and the categorisation scheme used, are appropriate and consistent with the evaluation results for the certified version.**

16.3 Evidence of assurance maintenance (AMA_EVD)

Objectives

The aim of this family of requirements is to establish confidence that the assurance in the TOE is being maintained by the developer, in accordance with the AM Plan. This is achieved through the provision of evidence which demonstrates that the assurance in the TOE has been maintained, which is independently checked by an evaluator. This check, termed an ‘AM audit’, is periodically applied during the lifetime of the AM Plan.

Component levelling

This family contains only one component.

Application notes

This family includes some evidence requirements that are similar to assurance requirements defined in the ACM, ATE and AVA classes. However, the AM audit does not require the evaluators to examine the evidence to the same extent as required by the components in these classes; rather, it requires a sampling approach to establish confidence that the assurance maintenance procedures are being followed correctly.

As part of the AM audit, the evaluators check (by sampling) that the configuration list and security impact analysis are consistent for the current version of the TOE, in terms of their identification of the TOE components that have changed from the certified version of the TOE.

AMA_EVD.1.3C requires the provision of evidence that the assurance maintenance procedures in the AM Plan are being followed. This covers all procedures referred to in AMA_AMP.1.11C, i.e. evidence of application of procedures relating to configuration management, maintenance of assurance evidence, performance of security impact analysis, and flaw remediation.

The evidence required in AMA_EVD.1.4C includes the provision of a list of identified vulnerabilities in the current version of the TOE. This is highlighted as a separate requirement because of the importance of ensuring, to a level consistent with the original evaluation assurance requirements, that the current version contains no security weakness that are exploitable within the TOE environment. The list in AMA_EVD.1.4C should include vulnerabilities arising from:

a) the developer’s analysis required by AVA_VLA.1, or higher component (if required for the certified version of the TOE);

b) any other reported security flaws handled by the flaw remediation procedures required by ALC_FLR.1(or ALC_FLR.2 if required for the certified version of the TOE).

AMA_EVD.1.5E requires the evaluators to confirm that functional testing has been performed on the current version of the TOE, and that the coverage and depth of testing is commensurate with the level of assurance being maintained. This check is performed by sampling the test documentation for the current version of the TOE.

AMA_EVD.1 Evidence of maintenance process

Dependencies:

AMA_AMP.1 Assurance maintenance plan

AMA_SIA.1 Sampling of security impact analysis

Developer action elements:

AMA_EVD.1.1D **The developer security analyst shall provide AM documentation for the current version of the TOE**.

Content and presentation of evidence elements:

AMA_EVD.1.1C **The AM documentation shall include a configuration list and a list of identified vulnerabilities in the TOE.**

AMA_EVD.1.2C **The configuration list shall describe the configuration items that comprise the current version of the TOE.**

AMA_EVD.1.3C **The AM documentation shall provide evidence that the procedures documented or referenced in the AM Plan are being followed.**

AMA_EVD.1.4C **The list of identified vulnerabilities in the current version of the TOE shall show, for each vulnerability, that the vulnerability cannot be exploited in the intended environment for the TOE.**

Evaluator action elements:

AMA_EVD.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

AMA_EVD.1.2E **The evaluator shall confirm that the procedures documented or referenced in the AM Plan are being followed.**

AMA_EVD.1.3E **The evaluator shall confirm that the security impact analysis for the current version of the TOE is consistent with the configuration list.**

AMA_EVD.1.4E **The evaluator shall confirm that all changes documented in the security impact analysis for the current version of the TOE are within the scope of changes covered by the AM Plan.**

AMA_EVD.1.5E **The evaluator shall confirm that functional testing has been performed on the current version of the TOE, to a degree commensurate with the level of assurance being maintained.**

16.4 Security impact analysis (AMA_SIA)

Objectives

The aim of the security impact analysis is to provide confidence that assurance has been maintained in the TOE, through an analysis performed by the developer of the security impact of all changes affecting the TOE since it was certified.

Component levelling

This family consists of two components, levelled according to the degree to which an evaluator validates the developer's security impact analysis.

Application notes

AMA_SIA.1 requires a sampling approach to validate the developer's security impact analysis. In some cases, AMA_SIA.2 may be preferred where a sampling approach is not considered sufficient to establish confidence that assurance has been maintained in the current version of the TOE, but where a formal re-evaluation is not considered necessary.

Both components in this family require the security impact analysis to identify all new and modified TOE components in the current version of the TOE (as compared with the certified version). The accuracy of this information is checked during either the associated AM audit (by sampling), or the associated re-evaluation of the TOE when the configuration list is checked under ACM_CAP.

AMA_SIA.1 Sampling of security impact analysis

Dependencies:

AMA_CAT.1 TOE component categorisation report

Developer action elements:

AMA_SIA.1.1D **The developer security analyst shall, for the current version of the TOE, provide a security impact analysis that covers all changes affecting the TOE as compared with the certified version.**

Content and presentation of evidence elements:

AMA_SIA.1.1C **The security impact analysis shall identify the certified TOE from which the current version of the TOE was derived.**

AMA_SIA.1.2C **The security impact analysis shall identify all new and modified TOE components that are categorised as TSP-enforcing.**

AMA_SIA.1.3C **The security impact analysis shall, for each change affecting the security target or TSF representations, briefly describe the change and any effects it has on lower representation levels.**

AMA_SIA.1.4C **The security impact analysis shall, for each change affecting the security target or TSF representations, identify all IT security functions and all TOE components categorised as TSP-enforcing that are affected by the change.**

AMA_SIA.1.5C **The security impact analysis shall, for each change which results in a modification of the implementation representation of the TSF or the IT environment, identify the test evidence that shows, to the required level of assurance, that the TSF continues to be correctly implemented following the change.**

AMA_SIA.1.6C **The security impact analysis shall, for each applicable assurance requirement in the configuration management (ACM), life cycle support (ALC), delivery and operation (ADO) and guidance documents (AGD) assurance classes, identify any evaluation deliverables that have changed, and provide a brief description of each change and its impact on assurance.**

AMA_SIA.1.7C **The security impact analysis shall, for each applicable assurance requirement in the vulnerability assessment (AVA) assurance class, identify which evaluation deliverables have changed and which have not, and give reasons for the decision taken as to whether or not to update the deliverable. These justifications shall be by reference to the documented changes affecting the security target, development or operational deliverables.**

Evaluator action elements:

AMA_SIA.1.1E **The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.**

AMA_SIA.1.2E **The evaluator shall check, by sampling, that the security impact analysis documents changes to an appropriate level of detail, together with appropriate justifications that assurance has been maintained in the current version of the TOE.**

AMA_SIA.2 Examination of security impact analysis

Dependencies:

AMA_CAT.1 TOE component categorisation report

Developer action elements:

AMA_SIA.2.1D The developer security analyst shall, for the current version of the TOE, provide a security impact analysis that covers all changes affecting the TOE as compared with the certified version.

Content and presentation of evidence elements:

AMA_SIA.2.1C The security impact analysis shall identify the certified TOE from which the current version of the TOE was derived, and shall identify all new and modified TOE components that are categorised as TSP-enforcing.

AMA_SIA.2.2C The security impact analysis shall, for each change affecting the security target or TSF representations, briefly describe the change and any effects it has on lower representation levels.

AMA_SIA.2.3C The security impact analysis shall, for each change affecting the security target or TSF representations, identify all IT security functions and all TOE components categorised as TSP-enforcing that are affected by the change.

AMA_SIA.2.4C The security impact analysis shall, for each change which results in a modification of the implementation representation of the TSF or the IT environment, identify the test evidence that shows, to the required level of assurance, that the TSF continues to be correctly implemented following the change.

AMA_SIA.2.5C The security impact analysis shall, for each applicable assurance requirement in the configuration management (ACM) and life cycle support (ALC) assurance classes, identify any evaluation deliverables that have changed, and provide a brief description of each change and its impact on assurance.

AMA_SIA.2.6C The security impact analysis shall, for each applicable assurance requirement in the delivery and operation (ADO) and guidance documents (AGD) assurance classes, identify any evaluation deliverables that have changed, and provide a brief description of each change and its impact on assurance.

AMA_SIA.2.7C The security impact analysis shall, for each applicable assurance requirement in the vulnerability assessment (AVA) assurance class, identify which evaluation deliverables have changed and which have not, and give reasons for the decision taken as to whether or not to update the deliverable.

Evaluator action elements:

AMA_SIA.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AMA_SIA.2.2E The evaluator shall check that the security impact analysis documents **all** changes to an appropriate level of detail, together with appropriate justifications that assurance has been maintained in the current version of the TOE.

Annex A

Cross reference of assurance component dependencies

The dependencies documented in the components of chapters 8-14 and chapter 16, are the direct dependencies between the assurance components. Table A.1 summarises both the direct dependencies and the indirect dependencies. The indirect dependencies are the cumulative result of iteratively including all the dependencies of each component identified as being a dependency.

Comp. Names	AUT	CAP	SCP	DEL	IGS	FSP	HLD	IMP	INT	LLD	RCR	SPM	ADM	USR	DVS	FLR	LCD	TAT	COV	DPT	FUN	IND	CCA	MSU	SOF	VLA
AUT.1-2		**3**	*1*												*1*											
CAP.1-2																										
CAP.3-4			**1**												**1**											
CAP.5			**1**												**2**											
SCP.1-3		**3**													*1*											
DEL.1																										
DEL.2-3		**3**	*1*												*1*											
IGS.1-2						*1*					*1*		**1**													
FSP.1-4											**1**															
HLD.1-2						**1**					**1**															
HLD.3-4						**3**					**2**															
HLD.5						**4**					**3**															
IMP.1-2						*1*	*2*			**1**	**1**							**1**								
IMP.3						*1*	*2*		**1**	**1**	**1**							**1**								
INT.1-2						*1*	*2*	**1**		**1**	*1*							*1*								
INT.3						*1*	*2*	**2**		**1**	*1*							*1*								
LLD.1						*1*	**2**				**1**															
LLD.2						*3*	**3**				**2**															
LLD.3						*4*	**5**				**3**															
RCR.1-3																										
SPM.1-3						**1**					*1*															
ADM.1						**1**					*1*															
USR.1						**1**					*1*															

Table A.1 - Assurance component dependencies[a]

Comp. Names	AUT	CAP	SCP	DEL	IGS	FSP	HLD	IMP	INT	LLD	RCR	SPM	ADM	USR	DVS	FLR	LCD	TAT	COV	DPT	FUN	IND	CCA	MSU	SOF	VLA
DVS.1-2															▨											
FLR.1-3																▨										
LCD.1-3																	▨									
TAT.1-3						*1*	*2*	**1**		*1*	*1*							▨								
COV.1-3						**1**					*1*								▨		**1**					
DPT.1						*1*	**1**				*1*									▨	**1**					
DPT.2						*1*	**2**			**1**	*1*									▨	**1**					
DPT.3						*1*	**2**	**2**		**1**	*1*							*1*		▨	**1**					
FUN.1-2																					▨					
IND.1						**1**					*1*		**1**	**1**								▨				
IND.2-3						**1**					*1*		**1**	**1**							**1**	▨				
CCA.1-3						**2**	*2*	**2**		*1*	*1*		**1**	**1**				*1*					▨			
MSU.1-3					**1**	**1**					*1*		**1**	**1**										▨		
SOF.1						**1**	**1**				*1*														▨	
VLA.1						**1**	**1**				*1*		**1**	**1**												▨
VLA.2-4						**1**	**2**	**1**		**1**	*1*		**1**	**1**				*1*								▨
AMP.1		**2**														**1**										
CAT.1		**2**																								
EVD.1																										
SIA.1-2																										

Table A.1 - Assurance component dependencies[a]

[a] In Table A.1, the left column represents groupings of specific components (using only the last three digits of the component name and an indicator of component number or range of numbers). Each non-empty box in the table indicates a specific component, identified by its name at the top of the column and the number in the box, on which the component in the left column is dependent. Bold numbers represent direct dependencies. Italicised numbers represent indirect dependencies. Dark shading represents the intersection of a component with itself. Dependencies from AMA components to assurance components are included in Table A.1, while AMA internal dependencies are shown in Table A.2 below. There are no dependencies from any non-AMA components to those in AMA, and so Table A.1 has no columns representing the AMA families.

AMA	A	C	E	S
Comp.	M	A	V	I
Names	P	T	D	A
AMP.1		**1**		
CAT.1				
EVD.1	**1**	*1*		**1**
SIA.1-2		**1**		

Table A.2 - AMA Internal Dependencies

DRAFT

Annex B

Cross reference of EALs and assurance components

Table B.1 describes the relationship between the evaluation assurance levels and the assurance classes, families and components.

Assurance Class	Assurance Family	Assurance Components by Evaluation Assurance Level						
		EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
Configuration management	ACM_AUT				**1**	1	**2**	2
	ACM_CAP	**1**	**2**	**3**	**4**	4	**5**	5
	ACM_SCP			**1**	**2**	**3**	3	3
Delivery and operation	ADO_DEL		**1**	1	**2**	2	2	**3**
	ADO_IGS	**1**	1	1	1	1	1	1
Development	ADV_FSP	**1**	1	1	**2**	**3**	3	**4**
	ADV_HLD		**1**	**2**	2	**3**	**4**	**5**
	ADV_IMP				**1**	**2**	**3**	3
	ADV_INT					**1**	**2**	**3**
	ADV_LLD				**1**	1	**2**	2
	ADV_RCR	**1**	1	1	1	**2**	2	**3**
	ADV_SPM				**1**	**3**	3	3
Guidance documents	AGD_ADM	**1**	1	1	1	1	1	1
	AGD_USR	**1**	1	1	1	1	1	1
Life cycle support	ALC_DVS			**1**	1	1	**2**	2
	ALC_FLR							
	ALC_LCD				**1**	**2**	2	**3**
	ALC_TAT				**1**	**2**	**3**	3
Tests	ATE_COV		**1**	**2**	2	2	**3**	3
	ATE_DPT			**1**	1	**2**	2	**3**
	ATE_FUN		**1**	1	1	1	**2**	2
	ATE_IND	**1**	**2**	2	2	2	2	**3**
Vulnerability assessment	AVA_CCA					**1**	**2**	2
	AVA_MSU			**1**	**2**	2	**3**	3
	AVA_SOF		**1**	1	1	1	1	1
	AVA_VLA		**1**	1	**2**	**3**	**4**	4

Table B.1 - Evaluation assurance level summary

Annex C

CC observation report (CCOR)

C.1 Introduction

The CC sponsoring organisations welcome feedback from the community and are particularly interested in observations and comments arising out of application of the criteria.

The CC sponsoring organisations have set up a body to coordinate and learn from the community experience and to ensure that future issues of the CC can benefit from that experience.

Comments, observations, and requests for interpretations should be sent to one of the addresses listed inside the front cover of the CC. If you require feedback on a specific evaluation matter, you should use the contact address which corresponds to the evaluation authority concerned.

C.2 Format of observation report

In order to allow for the automated categorisation of the observations, a standard observation format is needed.

The following provides a description of each structure of the required comment format and an example of a comment in the required format.

If you are submitting one or more observations by electronic mail or other machine readable format, you must use the ASCII text format to guarantee that your submission can be processed by an automated tool. You must also insert the tags defined below, each starting in the first column, as this will greatly assist in the automated handling of your input.

Each observation report should consist of three parts.

a) The first part consists of a tags **$1:** to **$4:**, which include the information to allow the unique identification of the originator. This first set of tags is required only once per single observation or batch of observations.

b) The second part consists of tags **$5:** to **$9:**, which include the information to allow the unique identification and categorisation of the observation, the actual observation itself and the suggested solution. The text of each observation should extend to as many lines as are needed to fully express the observation. There can be one or more observations in an observation report.

   The set of tags **$5:** to **$9:**, comprising this second part of the observation report, should be repeated for each observation being submitted.

c) The third part consists of a single terminating tag **$$:**. This final tag is required only once per single observation or batch of observations.

C.2.1 Tag definitions for observation report

Each tag must start at the first column of a new line.

$1: Originator name

The characters “**$1:**” without the quotation marks, followed on the same line by the name of the commenter (only required once per message).

$2: Originator organisation

The characters “**$2:**” without the quotation marks, followed on the same line by the originator organisation/affiliation (only required once per message).

$3: Return address

The characters “**$3:**” without the quotation marks, followed on the same line by the electronic mail or other address for response (only required once per message).

$4: Date

The characters “**$4:**” without the quotation marks, followed on the same line by the submission date of observation (only required once per message). The date should be formatted as:

YYMMDD

where YY refers to the last two digits of the calendar year, MM refers to the two digit representation of the month, and DD refers to the two digit representation of the day. For example, 29 December 1997 should be formatted as:

971229

and 5 January 1998 should be formatted as:

980105

$5: Originator report reference identification

The characters “**$5:**” without the quotation marks, followed on the same line by the reference for observation that is unique to originator. Please include your initials or similar unique discriminator, e.g. ABC1234.

$6: One line summary/title of observation

The characters “**$6:**” without the quotation marks, followed on the same line by the short summary/title for problem (up to 60 characters).

$7: CC document reference

The characters “**$7:**” without the quotation marks, followed on the same line by the single reference to the affected area of the CC as detailed as appropriate. The CC version for which the comment is being provided is required. Where possible, part number, section, paragraph, class, family, component, or requirement reference should be provided.

The template for CC document reference is as follows:

$7: Version / Part / Document Identifier / Keyword

The CC document reference template should be completed as follows (see below for completed example):

a) The characters "**$7:**" without the quotation marks, to indicate the start of an observation.

b) Identification of the Version. The CC Version can be found on the title page of each CC Part. It can also be found in the footer of every internal page within each Part. An example is:
Version 2.0

c) A "/" character, without the quotes, should be inserted between the Version and the Part identifiers.

d) Part:
Valid identifiers for the CC Part are:
P1 for Part 1
P1A for Part 1 Annex A
P1B for Part 1 Annex B
P1C for Part 1 Annex C
P1D for Part 1 Annex D
P1E for Part 1 Annex E
P2 for Part 2
P2A for Part 2 Annex A
P2B for Part 2 Annex B
....
P2N for Part 2 Annex N
P3 for Part 3
P3A for Part 3 Annex A
P3B for Part 3 Annex B
P3C for Part 3 Annex C

e) A "/" character, without the quotes, should be inserted between the Part and the Specific Document identifiers.

f) The Specific Document Identifier to which the comment applies in the CC. It should be as specific as is possible. The following list of options is provided in order of decreasing detail, such that if an option applies to your comment (when checking the options in order) then you should follow the directions within that option. If your comment applies to more than one of the options below, then you should consider following the directions in those additional options to determine other document identifiers and separate the resulting list of document identifiers with a comma.

If the comment refers to an element then the complete element identifier should be provided (e.g. FIA_ATD.1.1).

If the comment refers to a component then the complete component identifier should be provided (e.g. ADV_FSP.1). Additionally, any relevant page numbers could also be provided (e.g. 123-123).

If the comment refers to a family then the complete family identifier should be provided (e.g. FAU). Additionally, any relevant page numbers could also be provided (e.g. 123-123).

If the comment refers to a Figure or Table, the name of the section in which the item appears, followed by the name of the item (e.g. Annex A, Figure A.1).

If the comment refers to an item in a paragraph the name of the section in which the paragraph appears followed by a reference to the paragraph should be provided (e.g. Chapter 2 Definition of “TSF”).

If the comment refers to a section then the complete section identifier, preceded by the word “Section” should be provided (e.g. Section 3.1.1). Additionally, any relevant page numbers could also be provided (e.g. 123-123).

g) A “/” character, without the quotes, should be inserted between the Specific Document identifier and the Keyword (if a keyword is provided).

h) An optional keyword can be provided if the author of the CCOR feels it would be helpful.

$8: Statement of observation

The characters “**$8:**” without the quotation marks, followed on the same (or a new) line by the comprehensive statement of observation or query. This field can span several lines. It must contain the actual text of the observation. It should include specific reference to examples of the observation, where appropriate.

$9: Suggested solution

The characters “**$9**” without the quotation marks, followed on the same (or a new) line by the proposed solution or solution approach. This field can span several lines. It should include specific replacement text when possible.

$$: Terminating tag

The characters “**$$:**” without the quotation marks. This enables an automated handling system to determine the end of the batch of observations (only required once per batch of observations).

C.2.2 Example observations:

$1: A. N. Other
$2: PPs ‘R’ US
$3: another@ppsrus.com
$4: 980131
$5: ano.comment.1

$6: Presentation comment.
$7: Version 2.0 / P2 / FDP_ACF.1 / Italicise
$8: The operations in the component FDP_ACF.1 should be italicised.
$9: Italicise the operations.
$5: ano.comment.2
$6: Missing requirement for audit.
$7: Version 2.0 / P2 / FAU, pg. 336 /
$8: The first sentence of this paragraph is incomplete.
$9: The first sentence should include “imminent” violations.
$$: This is the end tag, the content is ignored.